(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

PROTOTYPING VISUAL INTERFACE

FOR

MAINTENANCE AND SUPPLY DATABASES

by

Henry Ray Fore

June 1989

Thesis Advisor: C. Thomas Wu

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) Code 52	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Prototyping Visual Interface for Maintenance and Supply Databases

12 PERSONAL AUTHOR(S)
Fore, Henry R.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1989 June	135

16 SUPPLEMENTARY NOTATION
The view expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES

FIELD	GROUP	SUB-GROUP
/		

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
object-oriented language, graphics interface, database language

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This research examined the feasibility of providing a visual interface to Standard Army Management Information Systems at the unit level. The potential of improving the Human-Machine Interface of unit level maintenance and supply software, such as ULLS (Unit Level Logistics System), is very attractive. A prototype was implemented in GLAD (Graphics Language for Database). GLAD is a graphics object-oriented environment for databases that gives novice and sophisticated users access to both data manipulation and program development through visual interaction. This thesis provided an extension to GLAD to demonstrate the ability to couple bitmap displays to database queries. [illegible] logistics [illegible] CSS (Combat support [illegible]); automation; Human computer interface [illegible]

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION
22a NAME OF RESPONSIBLE INDIVIDUAL Professor C. Thomas Wu	22b TELEPHONE (Include Area Code) (408)646-3391 22c OFFICE SYMBOL Code 52 Wq

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

U.S. Government Printing Office: 1986—606-24

Unclassified

PROTOTYPING VISUAL INTERFACE FOR MAINTENANCE AND SUPPLY DATABASES

by

Henry Ray Fore
Captain, United States Army
B.S., United States Military Academy, 1978

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
1939

Author: Henry R. Fore
Henry R. Fore

Approved by: [illegible]
C. Thomas Wu, Thesis Advisor

David K. Hsiao
David K. Hsiao, Second Reader

Robert B. McGhee
Robert B. McGhee, Chairman, Department of Computer Science

K.T. Marshall
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

This research examined the feasibility of providing a visual interface to Standard Army Management Information Systems at the unit level. The potential of improving the Human-Machine Interface of unit level maintenance and supply software, such as ULLS (Unit Level Logistics System), is very attractive. A prototype was implemented in GLAD (Graphics Language for Database). GLAD is a graphics object-oriented environment for databases that gives novice and sophisticated users access to both data manipulation and program development through visual interaction. This thesis provided an extension to GLAD to demonstrate the ability to couple bitmap displays to database queries.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my advisor Dr. C. Thomas Wu for his patience, assistance and understanding in completing this research work.

I also thank my beautiful and most understanding wife, Tae Sil, for her constant encouragement and support through the long course of obtaining this masters.

I. INTRODUCTION

A. BACKGROUND

1. Automation in Army Logistics

The United States Army's Combat Service Support (CSS) automation architecture has undergone dramatic changes in the last ten years to meet AirLand Battle requirements and to meet head on the influx of new and complex logistics systems. Three major systems that are planned or currently being fielded are the Corps and Theater Automatic Data Processing Service Center (CTASC-II), The Army Combat Service Support Computer System (TACCS) and the Unit Level Computer (ULC). CTASC-II, a minicomputer, will provide automation to support logistics, medical and personnel systems at the corps and echelons above corps. TACCS is a modified Burroughs B-26 microcomputer configured to operate as a stand-alone communications capable, multiuser system [Ref. 1].

TACCS is designed to run all Standard Army Management Information Systems (STAMIS). STAMISs are the software systems that manage all major supply and maintenance databases for the army. TACCS will be found primarily at direct and general support activities. The ULC will be a commercially equivalent microcomputer consisting of off-the-shelf components in order to stay competitive with technological advances. The ULC is designed to perform the maintenance and transportation functions at the battalion and unit level. The ULC at the present time is under procurement, however, the Zenith 248 is being fielded to run its Unit Level Logistics System (ULLS) prototype software.

Additionally, the Army is exploring the technology of storing its thousands of technical manuals onto electronic storage media embedded inside of interactive capable software [Ref. 2]. Three projects the Army is currently working on are the Miniaturized Electronic Information Delivery System (MEIDS), the Electronic Maintenance Publication System (EMPS) and the Personal Electronics Aid for Maintenance (PEAM). Their commitment to these projects stems from the need to reduce the dependence on our bulky paper technical manuals and improve maintenance productivity. Compact discs, videodiscs and mass storage cartridges are the primary electronic storage media being tested.

The objective of this automation modernization is to uncluster and organize our combat service support activities into paperless and highly efficient organizations. We as managers welcome the changes, considering the continued procurement of sophisticated equipment accompanied by voluminous and complex repair parts technical manuals. However, as we continue to automate database systems at lower level unit activities, we are introducing computers to non-computer literate individuals. Our concern, which is the focus of this thesis, is to consider the human-computer interaction aspects of current army database management systems (DBMS) to support non-computer literate users.

2. **ULLS**

ULLS is a software system designed to automate maintenance and supply functions at the unit level. The Prescribed Load List (PLL), a repair parts management system, and The Army Maintenance Management System (TAMMS) are two major functions that are automated in ULLS. The latest version of ULLS will provide for timely equipment status information and improve the managing of resources and readiness rates. The bulk of these two systems are databases of a unit's equipment, repair parts and historical data. Currently, ULLS-II is written in Informix's SQL DBMS. SQL (pronounced Sequel) is a commercial relational database query system that contains other features such as database modification and security constraint specification capabilities. ULLs running under SQL is presented to the user through an intensive hierarchial menu system and no graphic capabilities.

B. THE NEED FOR RESEARCH

The fielding of ULLS provides us the opportunity to revalidate our human-machine interface design goals. Even though menu systems are an acceptable form of human-computer interaction for database queries on a microcomputer, visual interaction through direct manipulation is the next logical step towards ease of learning and ease of use of our logistical and maintenance DBMSs.

What do we mean by direct manipulation? We refer to direct manipulation as a user interface design that employs the following principles [Ref. 3]:

- continuous representation of the objects and actions of interest
- physical actions or labeled button presses instead of complex syntax
- rapid incremental and reversible operations whose impact on the object of interest is immediately visible.

The visibility and direct manipulation of the objects of interest are the central ideas behind the above mentioned principles. Some examples of direct manipulation systems are full- screen display editors, computer-aided design/manufacturing programs and video games. All of these examples give the user the feeling of control over the application which elicits enthusiasm, speeds learning and aids in retention. Given the diversity of users, direct manipulation of logical objects is appealing to novices, easy to remember for intermittent users and rapid for frequent users. [Ref. 3] We feel that graphics coupled with this design interface would make army logistical software, such as ULLS, more receptive to the end user.

The focus of this research tested the feasibility of providing this such graphic support. The specific objectives of this research were:

Bitmap display extension to Graphics Language for Database (GLAD). This powerful capability gives the designer the capability to store pictorial graphics as attributes in tuples and display them in a windows environment simultaneously with their data.

- A graphics interface to access existing DBMS.

C. THESIS ORGANIZATION

Chapter II provides a brief discussion on the object-oriented programming language ACTOR and its software development environment.

Chapter III explains the implementation details of the prototype. Graphics Language for Database (GLAD) background and the reason it was chosen as the interface for use in the prototype is discussed.

A sample session of the research results running in GLAD is presented. The various windows that can be manipulated are illustrated and described. This chapter also explains the implementation of the bitmap displays and associated implementation difficulties.

Chapter IV follows with possible enhancements as continuation of research in this area. Benifits and final discussion of the research are also presented.

II. OBJECT ORIENTED PROGRAMMING WITH ACTOR

A. ELEMENTS OF OBJECT ORIENTED PROGRAMMING

OOP is coming of age with the introduction of high speed parallel processors, sophisticated software development windowing environments, and the desire for rapid prototyping of software systems. Although the term OOP has been used differently, the fundamental principles of OOP design are the same. One formal definition is:

Object-oriented design is the construction of software systems as structured collections of abstract data type implementations [Ref. 4:p. 59].

The basic principles of OOP are encapsulation, reusability, late binding and information hiding. We find many procedural languages claiming to be object-oriented languages, however, for a language to really be considered object-oriented in design, the language should keep the following steps toward 'object-orientedness' [Ref. 4: p. 60]:

- Object based modular structure: Systems are modularized on the basis of their data structures.
- Data abstraction: Objects should be described as implementations of abstract data types.
- Automatic Memory Management: Unused objects should be deallocated by the underlying language system, without programmer intervention--garbage collection.
- Classes: every non-simple type is a module, and every high-level module is a type.
- Inheritance: A class may be defined as an extension or restriction of another.

- Polymorphism and late binding: Program entities should be permitted to refer to objects of more than one class, and operations should be permitted to have different behavior in different classes.

- Multiple and repeated inheritance: It should be possible to declare a class as heir to more than one class.

We chose to use one such language that possesses most of the above features called Actor.[1] Actor is an object oriented progamming language (OOL) for MS-DOS microcomputers from The Whitewater Group Inc. Actor satisfies all of the above criteria except multiple inheritance. Actor uses singles inheritance. Actor, as an OOL, integrates three fundamental concepts found in most true OOLs: objects, classes, messages and methods. Before we go any further, we will briefly explain these concepts.

1. **Objects**

We find the most fundamental concept that conventional programmers have difficulty grasping is that of an object. Intuitively, objects in the physical world are airplanes, banks, employees, pictures, payrolls, software programs, and so on. The goal of OOP is to represent these physical objects, as we would naturally think of them, as computer representations. An object in the computer domain consists of some private data and a set of operations that can operate on that data. The object will perform one of its operations only when requested to

[1]Actor is a trademark of The Whitewater Group Inc.

do so by sending a request or message to that particular object. The receiver of that message then responds by choosing some operation that implements the name of the message. The operation is then executed and control returned to the caller. [Ref. 5:p. 50] Everything in Actor is an object. By "everything," we mean numbers, characters, arrays, windows, and so on - everything is an object [Ref. 6: p.35].

2. Classes

Since we have an idea of what comprises objects, the next most important concept of OOL is classes. A class describes the patterns or structures common to categories of objects, such as: the class of tanks, of polygons, of students, and so on. Any class describes a set of objects, where these objects are called instances of a class.

As we described in the description of objects, each object (instance of a class) has its own private data. We call this private data instance variables in Actor. Instance variables are only accessible by an instance of a particular class where they are defined. Other objects may 'see' values of the instances of a class only by sending a message to that class. Actor stores these instance variables in a variable dictionary, one for each defined class. Actor has over 100 predefined classes with about 80 loaded into the system for software development. [Refs. 7,8]

3. **Methods and Messages**

Now that we understand the ideas behind objects and classes, the final concept is that of **methods** and **messages**. The set of operations, that were described earlier as being part of objects, are formally referred to as **methods** in Actor. Methods are the implementations of the operations found in instances of a class. We find no functions or procedures in Actor. Functions and procedures are replaced by methods. The method is the place where all the work gets done.

How does the code get executed in Actor? It gets executed by sending **messages** to an object. A **message** in Actor consists of a **selector**, a **receiver** and a list of arguments. When you send a message to the object, it looks to see if a method of the same name as the selector exists, if so, it executes the method. If the method does not exist within the instance of that class, an error message is generated. Method names need not be unique. The syntax of a method definition in Actor is:

```
/* Comments */
Def <methodName>(self [argument list [| <local
variables>]])
( statement 1;
  statement 2;
      .
      .
      .
  statement n;
 ( !!
```

Actor methods can take up to eight arguments, that can not be changed, and up to eight local variables. The local variables can be assigned and exist only during the life of the method. The above method can be invoked by sending it a message:

```
methodName (Receiver, arguments);
```

The selector would be "methodName," the receiver is "Receiver" and the list of arguments "arguments."

B. INHERITANCE

One of the most powerful features of OOLs is inheritance. Why reinvent something that already exists? Why not extend on something that already works? Why not define new classes in terms of old classes? Inheritance permits us as programmers to affirmatively implement the issues raised by these questions in OOP. The following are the fundamental ideas behind inheritance [Ref. 4:p. 217]:

- avoids rewriting and duplication of code.
- reduces errors and inconsistencies.
- captures commonalities between classes.

In Actor, inheritance involves the hierarchial inheriting of behavior between classes. Key terminology used here are words like **ancestors**, **descendants** and **class trees**. Actor's class tree is the hierarchial ordering of all its predefined classes. Each class inherits behavior from classes above it, its ancestors, and passes down behavior to classes below it, its descendants. By behavior, we mean, a class has access to all of its own instance variables and methods, as well as, to those of its ancestors in the class tree. Actor uses **single inheritance.** Each class is permitted only one direct ancestor. [Ref. 6: p. 39]

The design of Actor's class tree reflects this inheritance. The most generic classes are at the top and the more specialized classes at the bottom. We can define additional classes and attach them to the class tree according to ancestor/descendant protocol. The classes we define must be descendants of classes already defined in Actor. However, we can define as many descendants of our user defined classes as we like. Actor is very powerful in this aspect. We can expand the class tree as large as we like. Of course, the intent of inheritance is to take advantage of the idea of reusability. Therefore, the design of an application should be well thought out to use the behavior of classes already defined.

C. POLYMORPHISM

Another important concept in Actor is that of **polymorphism.** Polymorphism literally means 'the ability to take several forms'. In OOP, it refers to the ability of a message or entity to react at run-time to several classes. Actor allows different classes to have the same method names. The methods could also have totally different implementations. The net effect of this is that we can send the same message to different objects to produce different responses. Polymorphism allows us to do this without conflicts. Further information on Polymorphism can be found in [Ref. 4: p. 224-227].

D. SOFTWARE DEVELOPMENT ENVIRONMENT

Now that we have explained the essence of OOP and Actor, let's briefly look at Actor's programming environment. As mentioned earlier, Actor is an object-oriented programming language (OOP), however, it is a Microsoft Windows (MS-Windows)[2] application. MS-Windows is an operating environment that provides a visual interface to users. This "visual interface" is essentially an extension of MS-DOS's operating system. MS-Windows runs under MS-DOS version 2.0 or higher on IBM[3] personal computers or compatibles. The visual interface is presented to us in the form of windows that contain graphic representations of user input, input options and system output, Figure 2.1. [Ref. 9:p. 4] Menus are the principal means of presenting the user options within an application. MS-Windows also provides multitasking capability; several applications or programs can be run simultaneously.

The concept of OOP, generically, applies to MS-Windows. All windows are objects that receive messages to perform operations on themselves. MS-Windows provides the

[2]MS and Microsoft are trademarks of Microsoft Corporation.

[3]IBM is a trademark of the IBM corporation.

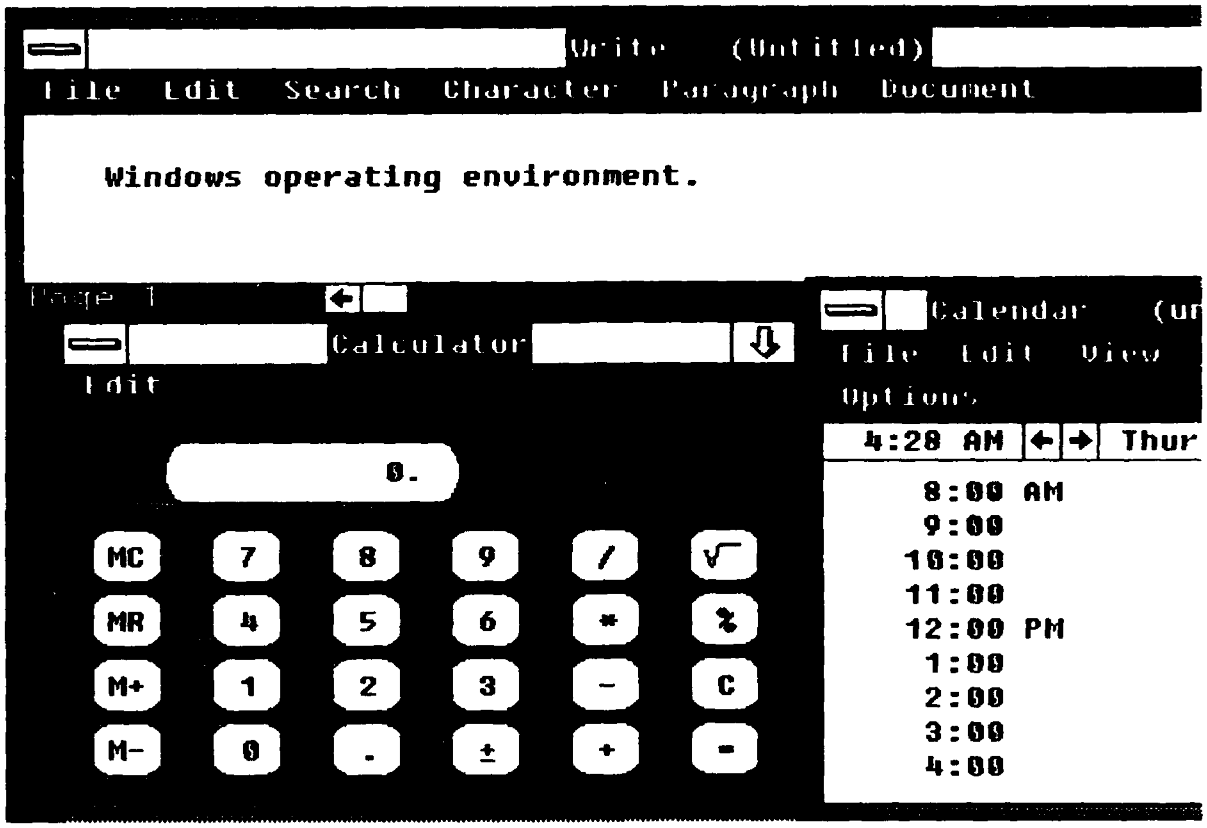


Figure 2.1 Sample Microsoft Windows Screen

programmer over 600 "Kit routines" to develop application programs to run under its window environment. Actor uses these Kit routines to design its user interface for its software development environment. Since Actor runs under MS-Windows, it is a MS-Windows application. Unlike other Window applications, Actor's source code is in the form of predefined classes. This Actor source code is available for our use as programmers. Since Actor runs under MS-Windows, it can also access and use any of MS-Windows functions to create menus, dialogue boxes, and so on, for Actor applications.

Actor provides the programmer a complete programming environment. Actor version 1.2 requires 640K RAM, a hard disk, a Graphics display and adapter and a mouse or other pointing device. [Ref. 6:p. 1] All adapters are supported via MS-Windows. This thesis research was tested successfully on a Department of Defense's standard contract Zenith 248 microcomputer in its standard configuration.

For software development, Actor offers us Work Space Windows, Browser Windows, Editor Windows, Debugger Windows, Inspector Windows and numerous other tools. There are numerous menu options within these various windows to aid us in our object-oriented tasks. Some include: object code templates, garbage collection, current static and dynamic availability, object inspection, object source code editors and others. These windows and their menu options in conjunction with dialogue boxes, icons, popup windows and error boxes makes Actor a powerful OOP language.

E. SUMMARY

We attempted in this chapter to briefly introduce some of the key terminology used in OOP. Objects, classes, messages and methods are the fundamental building blocks in OOP. Actor is considered a pure OOP language because [Ref. 6:p 35]:

- **EVERYTHING** in Actor is an object. Numbers, characters, arrays, strings, windows, methods, and so on - are objects.
- Every action that occurs in Actor (except for calling MS-Windows or MS-DOS) is the result of sending a message to an object, which responds to it by executing a method.

Inheritance in conjunction with polymorphism gives an OOL its power. Classes can inherit behavior from its ancestors and pass its behavior to its descedants. Polymorphism allows instances of classes to have identical method names. This means that messages of the same name can be sent to different objects at run time to produce different results without harm. Actor runs under a MS-Windows and provides the programmer its source code to develop applications under a user-friendly software development environment.

III. IMPLEMENTATION

In this chapter, we will look at the application we used to demonstrate the merits of this research. We will discuss why we used GLAD; show GLAD in operation through a sample session with a sample database; and give the implementation details of displaying bitmaps in GLAD.

A. GLAD

The basis of this research was to explore the possibilities of providing a graphical interface to current army database systems. Instead of developing a prototype system from scratch, we enforced the principle of reusability and used Graphics LAnguage for Database (GLAD).

1. Background

GLAD is an ongoing project developed by Professor C. Thomas Wu at the Naval Postgraduate School in Monterey, CA. GLAD is a prototype for visual interfaces to databases, supporting high-level semantic data models. The motivation for GLAD stems from the realization that an end-user visual interaction tool was needed for database systems. Irrelevant of the type of database system (relational, network, hierarchical), GLAD will provide the end-user a coherent interface, through which he or she can visually interact with the system for data manipulation and program development. [Ref. 10:p. 2] Previous thesis students have completed development of GLAD's top level window, data manipulation language (DML) and data definition language (DDL). Current work includes implementation of Help systems and development of a backend link to an actual database system.

`. Environment

We covered in Chapter Two the fundamentals of object oriented programming and Actor. GLAD was developed using the Actor programming language. The primary advantage of using Actor as the implementation langauge for GLAD is that it served as a rapid prototyping tool. Just as rapid prototyping was important in the development of GLAD, so it is in this research project. The intent was not to build a complete system, but only a system to demonstrate the feasibility of implementing our goals. Actor, as a rapid prototyping tool, gave us the flexibility to quickly communicate interface design alternatives and provide for rapid design changes. See [Ref. 10] for a more detaile discussion of the advantages of using Actor to implement GLAD. GLAD as a stand alone application has the same hardware and software requirements as the Actor programming language. A sample GLAD session will be given in Section B of this chapter.

3. Why GLAD?

The original idea for this thesis came about from a sister project called ARGOS. ARGOS is another project at the Naval Postgraduate School being supervised by Professor C. Thomas Wu. ARGOS involves putting an entire surface ship's repair parts database system ontc a Macintosh microcomputer using HyperCard technology. Once this database is loaded, the end-user will be able to select, manipulate and requisition parts using the system's graphics interface package.

The ultimate goal of this thesis is that of ARGOS's, provide a graphical user interface to logistical databases. The problem we faced was that HyperCard technology, a powerful tool for graphics generation, is for Macintosh machines only. Considering that the Army uses IBM compatible microcomputers, the conveniences provided by HyperCard were not available. GLAD turned out to be our best candidate for transferring the ARGOS idea onto IBM hardware.

GLAD was already developed to the point where minor modifications were needed to input a sample army repair parts database. The difficult part of the implementation was the storage and display of bitmaps inside GLAD. This feature had to be added in order to display pictures of repair parts from technical manuals and other publications. The bulk of this research went into providing this extension to GLAD.

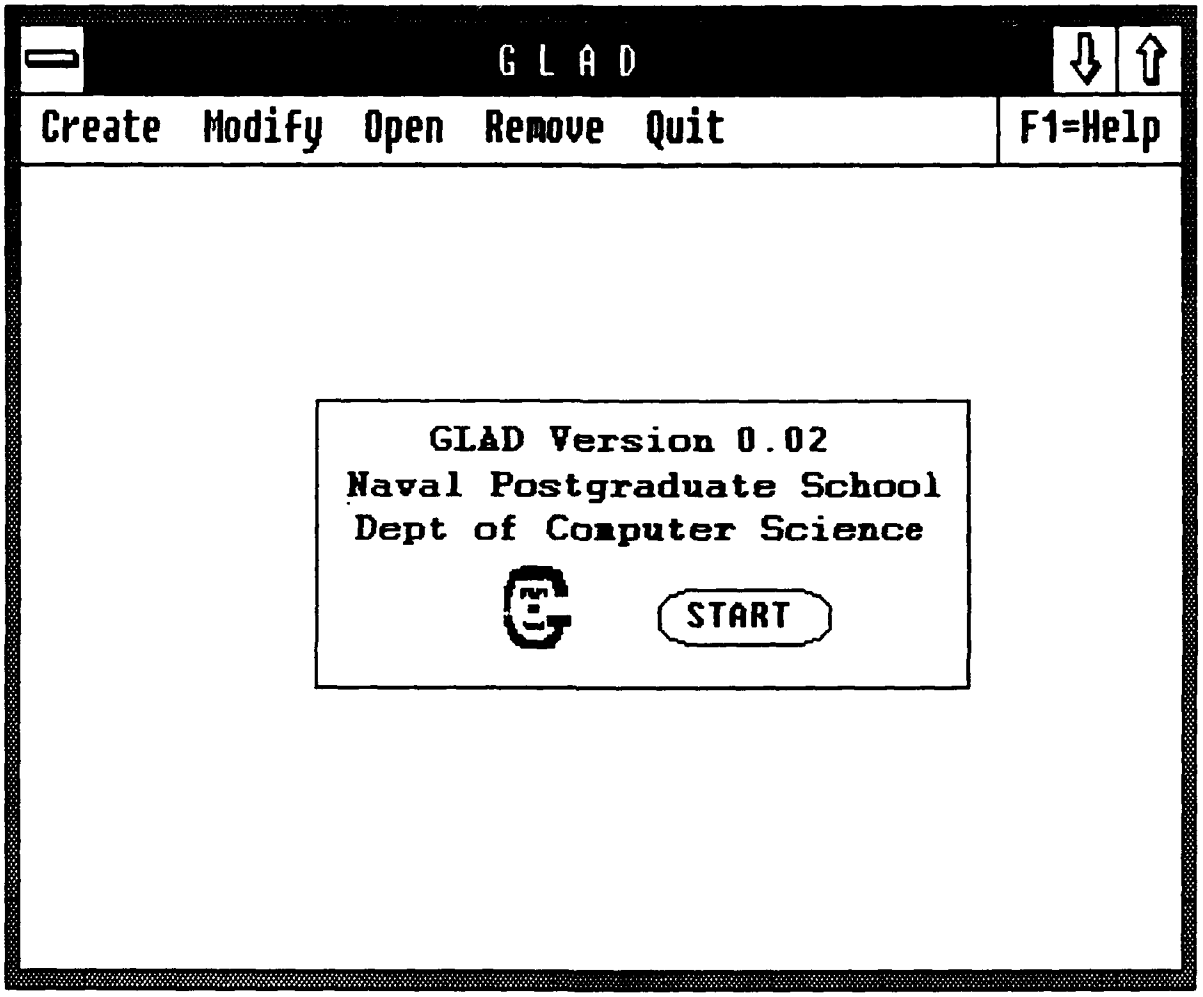
B. SAMPLE SESSION

The following is a sample run of GLAD with a sample database and bitmap extension implemented. This particular session was run on an IBM PS/2 Model 60 microcomputer with VGA monitor, 1 MB of RAM, 40 MB hard drive, a mouse, Actor version 1.2, and MS-Windows versions 2.03. Only those major features of the system that demonstrate the results of the research will be illustrated. Additionally, only the interface side of GLAD will be explained. The database query details or associated logic will not be covered in the scope of this presentation. See [Refs. 8, 11] for detail explanations of the data manipulation and data definition languages, respectively.

We used the components of a U.S. Army Attack Helicopter, Model 1S(AH1S) as the sample database. The AHIS is sometimes referred to as a "Cobra" attack helicopter. This database is representative of the repair parts that could be found in any of the army's maintenance or supply databases. Representative repair part groups from Technical Manual 55-1520-221-23P were chosen to demonstrate the capabilities of GLAD as a visual interface to a database [Ref. 12]. The contents of the schema file and data file are at Appendices C and D, respectively. Detailed implementation procedures will be discussed in the next section.

1. Main GLAD Window

The GLAD application consists of a top level window that serves as the gateway to selecting, creating, modifying or removing databases. Figure 3.1 shows GLAD's main window along with a dialogue box to start the system. The user always has several options of moving around in the windows and making selections. The primary and most



GLAD Version 0.02
Naval Postgraduate School
Dept of Computer Science

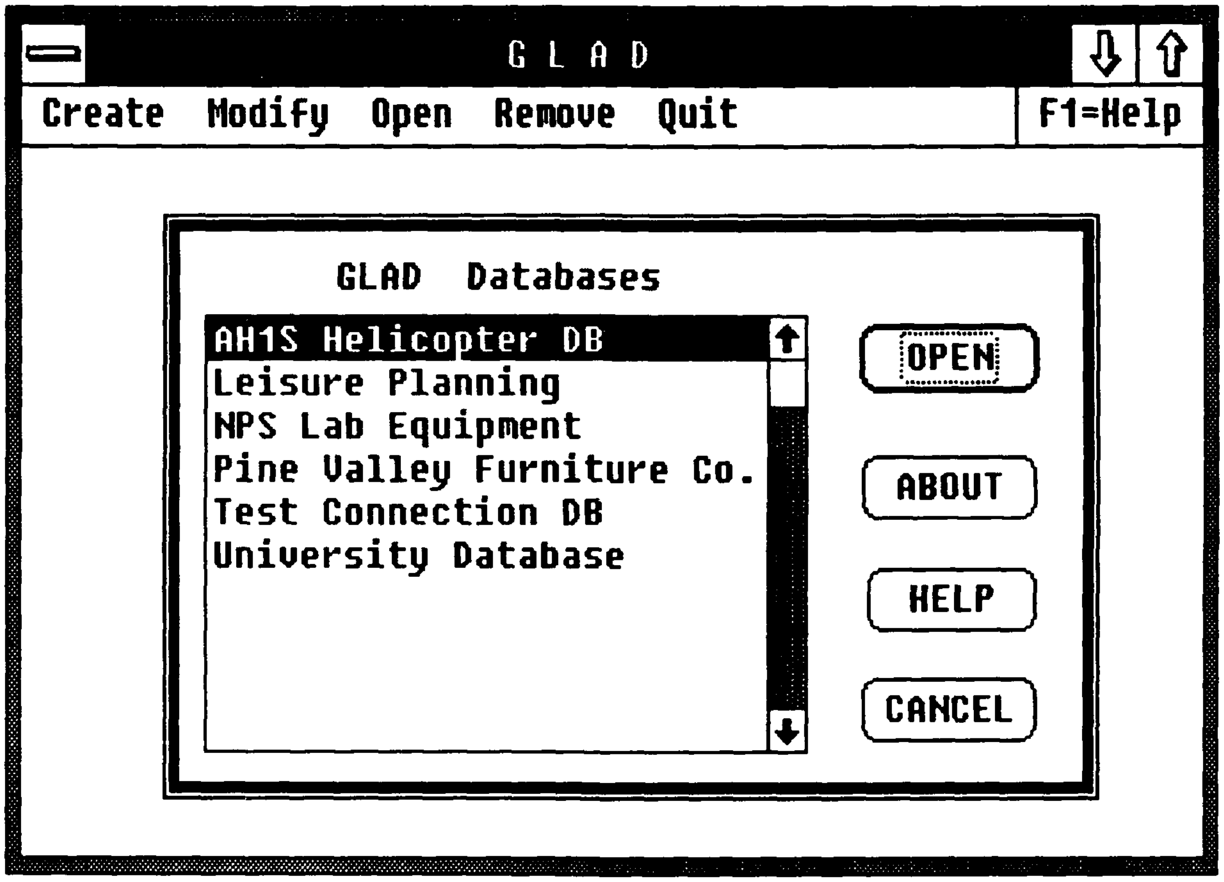
Figure 3.1 GLAD Main Window

commonly preferred method is the mouse. The system is started by either clicking on the start button with the mouse or hitting the enter/return key on the keyboard.

The caption bar contains a menu of options. All of GLAD's windows have the self-explanatory **Help** and **Quit** options. The **Help** systems, developed by another thesis student, are presented in separate popup windows that include extensive text and graphics. A user can exit any window by selecting **Quit**, double clicking on the window's system box in the upper left hand corner or by selecting **Close** on the window's pull down menu. **Help** and **Quit** will not be discussed further in future menu option descriptions.

A programmer or database administrator has several menu options at their exposal to perform database management functions: **Create**, **Remove** and **Modify**. The **Create** option creates new databases. The **Remove** option allows for the removal of databases. The **Modify** option allows for the modification of databases. The typical end-user would not have access to these particular menu options. Restricted access features are planned enhancements.

The **OPEN** option is available to all users who have access to the system. By clicking the mouse on this option, the user is presented a dialogue box listing the various databases available in the system, Figure 3.2. Appendix B contains the data files currently used in GLAD.



AH1S Helicopter DB
Leisure Planning
NPS Lab Equipment
Pine Valley Furniture Co.
Test Connection DB
University Database

Figure 3.2 GLAD Available Databases Dialogue Box

Notice that this window can be scrolled to display databases that may not fit in the window. The user is again presented with options, this time in the form of push buttons. A database selection must be highlighted before it can be opened. The user highlights a particular database in the GLAD Databases dialogue box by moving the mouse to and clicking on the desired line, "AH1S Database" in this case. The user then clicks the mouse on the **Open** button. The selected database and its Data Manipulation Window is then opened.

2. DML Window

The Data Manipulation Language (DML) window illustrates the power of human-computer visual interfaces, Figure 3.3. What the end-user sees is what the end-user gets.

a. Object definition

The DML window gives the user manipulation of the database schema and its data. Since GLAD is based on an object-relationship model, objects (entities) of the database schema are shown as rectangular boxes in the DML window. FLT_CTL (flight control system), ENV_CTL (environment control system), AF_TOOLS (airframe tools), FUEL_SYS (fuel system) and ELEC_SYS (electrical system) are all objects of the attack helicopter database schema, see Appendix C. These objects are essentially the primary component system groups of an attack helicopter.

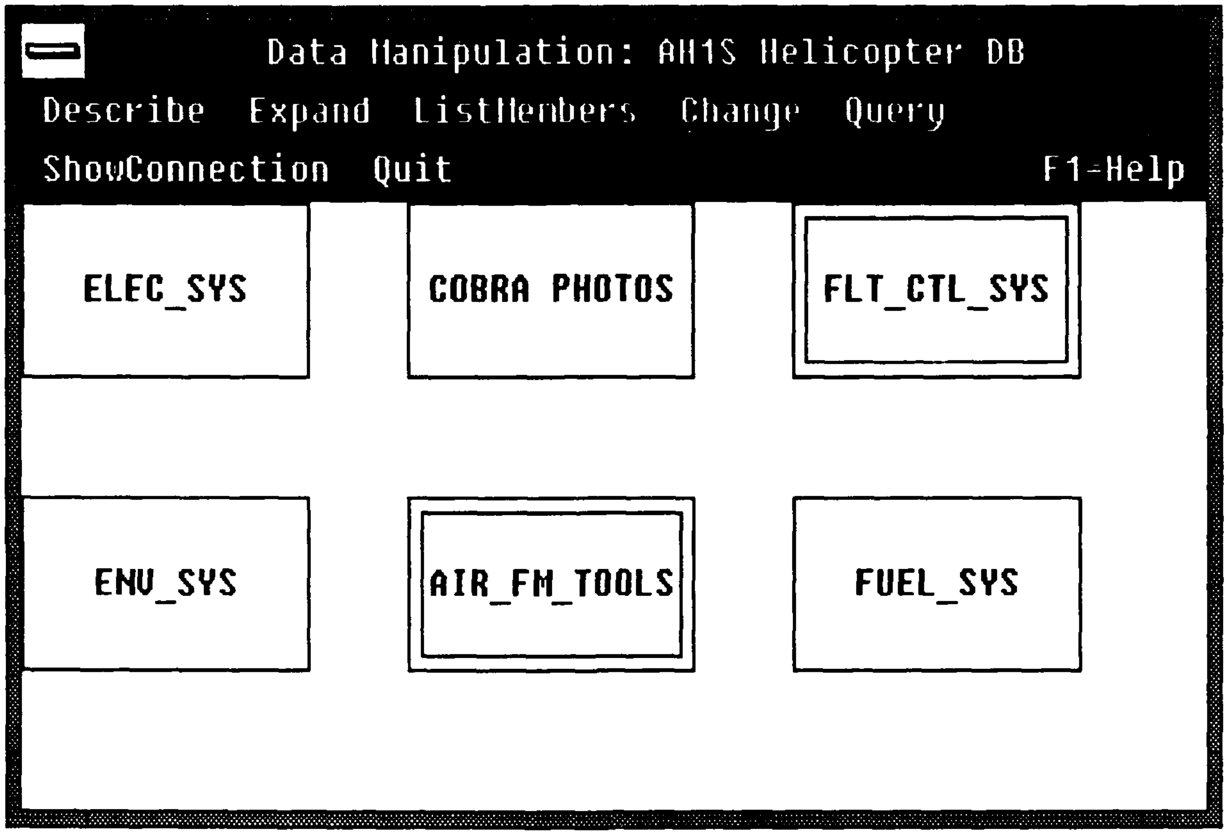


Figure 3.3 GLAD DML Window

The user selects a helicopter component system, an object, by clicking on it with the left mouse button. The color of the rectangle will change to indicate that it is the currently selected object. Once selected, operations can be performed on the object with various menu options or even repositioned within the window. Subgroups or subclasses within the primary groups are represented visually as rectangles within larger rectangles. We call this 'nesting' of objects. Nesting of objects are based on the generalization principles of databases theory.

b. Expand

Nested objects within the DML window can be expanded to display the more specialized subgroups. Once a nested object is selected the user can click on the **EXPAND** menu option. A new nested schema window showing the nested objects is then displayed. Figure 3.4 illustrates the expansion of the FLT_CTL system. Notice that the user is presented the same menu options, except for ShowConnection, as the main DML window. Even though only one window is shown in Figure 3.4, multiple nested windows to any level are possible. The Nested objects represents ISA hierarchy structures used in depicting generalizations. [Ref. 13] Since most technical manuals are made up of many subgroups or subclasses of more generalized components, nesting of objects is a highly desirable interface feature.

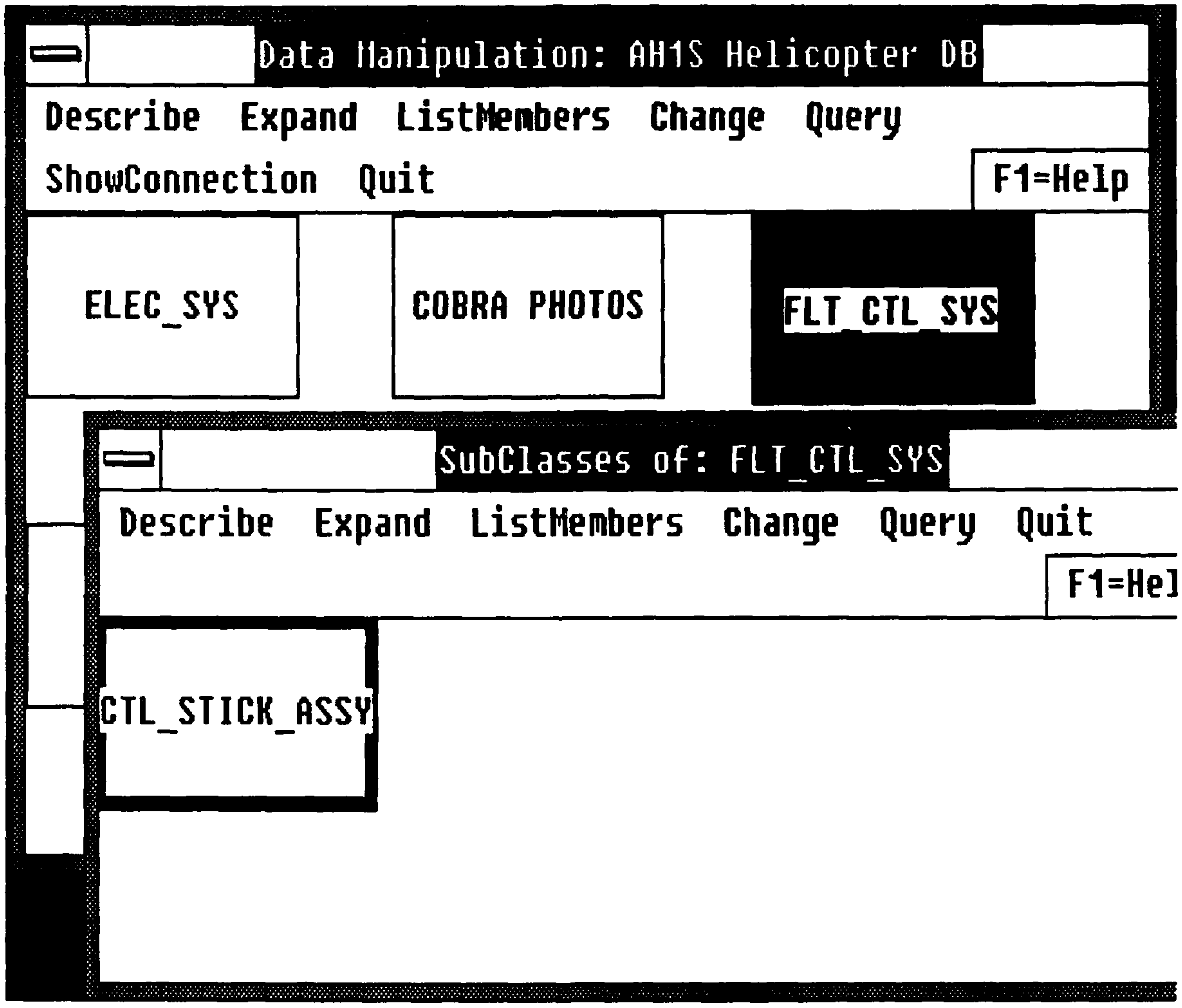

Data Manipulation: AH1S Helicopter DB
Describe Expand ListMembers Change Query
ShowConnection Quit
F1=Help
ELEC_SYS
COBRA PHOTOS
FLT_CTL_SYS
SubClasses of: FLT_CTL_SYS
Describe Expand ListMembers Change Query Quit
F1=Hel
CTL_STICK_ASSY

Figure 3.4 Nested DML Window

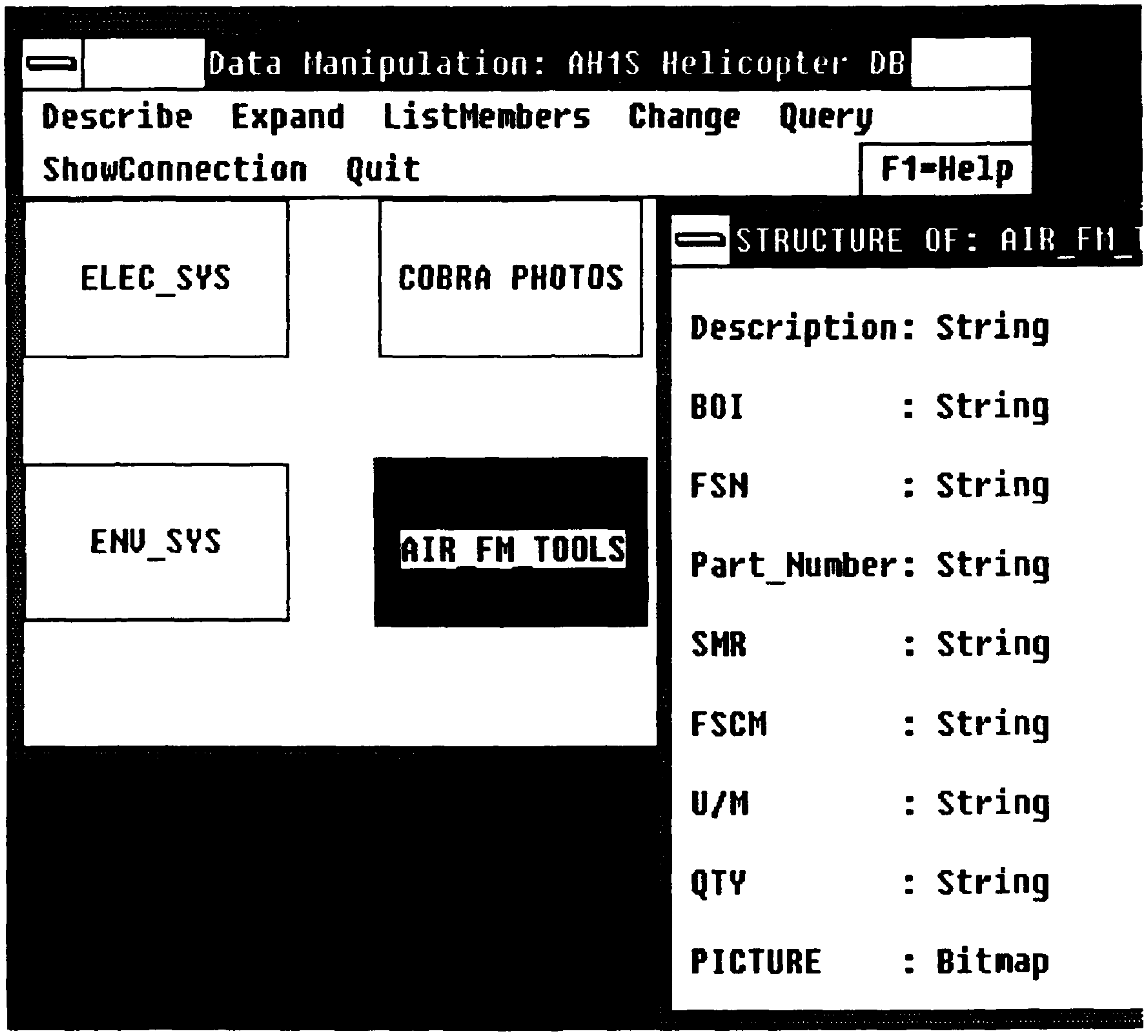
c. Describe

Back at the DML window, Figure 3.3, **Describe** is another important menu option. This menu option lets the user understand more about the schema by viewing the relational information of the selected object. Figure 3.5 shows the schema for the fuel system. Again, multiple **Describe** windows are permitted.

d. ListMembers

The DML menu options examined up to now dealt with viewing of schema information. The **ListMembers** option is a 'pull down' menu option that permits the user to view and manipulate the data of the database. The selections available from **ListMembers** are **All at Once** and **One by one.**

The "All at Once" option gives the user the instances of a selected object all at one time, Figure 3.6. Selecting the **All at Once** menu option displays a **ListMembers** window containing all attributes of the selected object and their associated tuples (data). The user can 'browse' through the tuples listed in the window by holding down the left mouse button and 'dragging' its cursor up and down the screen. This method enables the user to highlight a specific tuple for modification or detailed viewing. Alternatively, scroll bars are provided on the right side and bottom of the window to scroll through the tuples than can not be displayed. As shown in



```
Description: String
BOI        : String
FSN        : String
Part_Number: String
SMR        : String
FSCM       : String
U/M        : String
QTY        : String
PICTURE    : Bitmap
```

Figure 3.5 Description Window

Figure 3.6, all the flight control system parts are listed as exactly as they are represented in the technical manual.

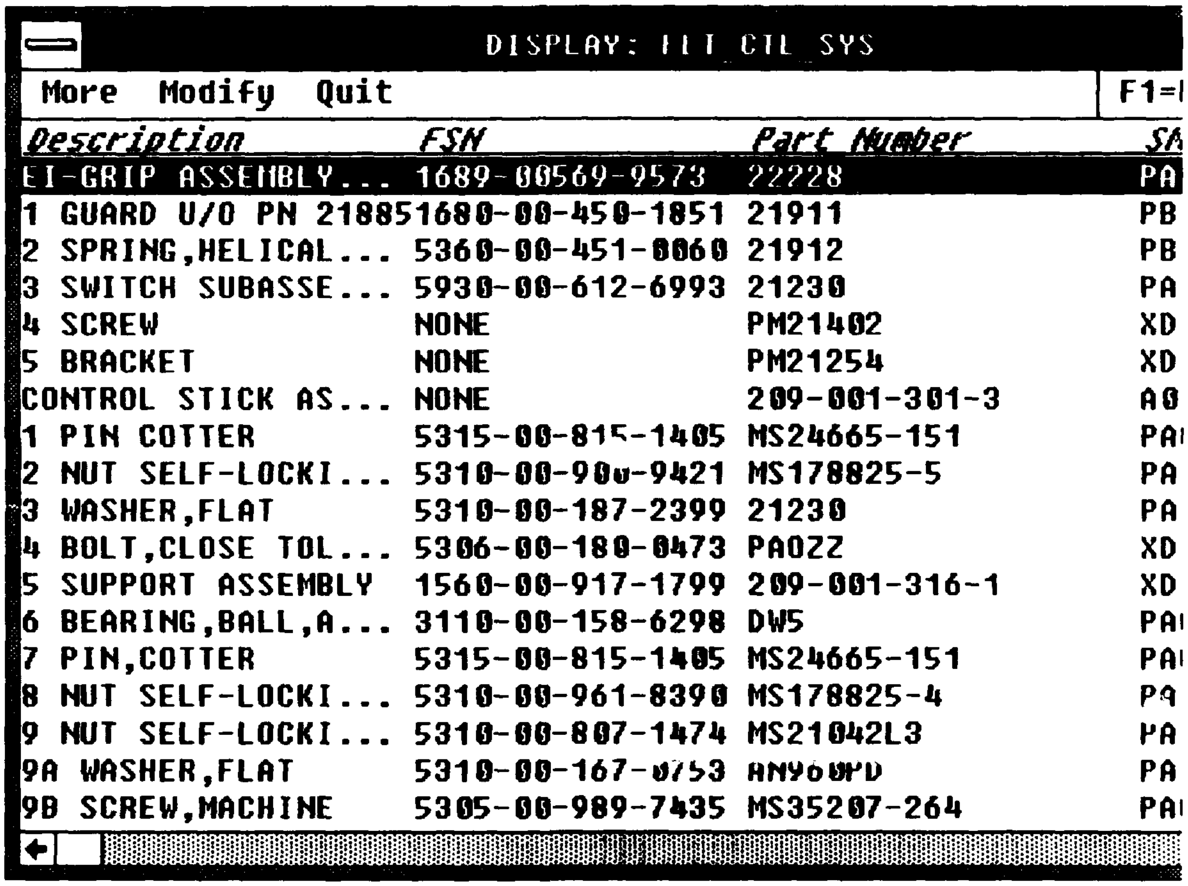
If the user wanted to see a more detailed format of an highlighted component, they would only have to select the **More** option in the "All at Once" window. By clicking on the **More** option, a DisplayOneWindow window would appear with more detailed information on the highlighted tuple, Figure 3.7. More on the specifics of this window in the next section. The **More** option is essentially displaying a single instance of tuple, which also a **One by one** option of **ListMember** DML window.

3. **DisplayOneWindow**

One by one on the DML window **Listmember** option displays instances of the selected objects one at a time. Selection of this option presents the user with the same DisplayOneWindow as you saw with the **More** option in **All at Once**. Our data is now displayed for the selected object in the DisplayOneWindow beginning with the first tuple in the database, see Figure 3.8.

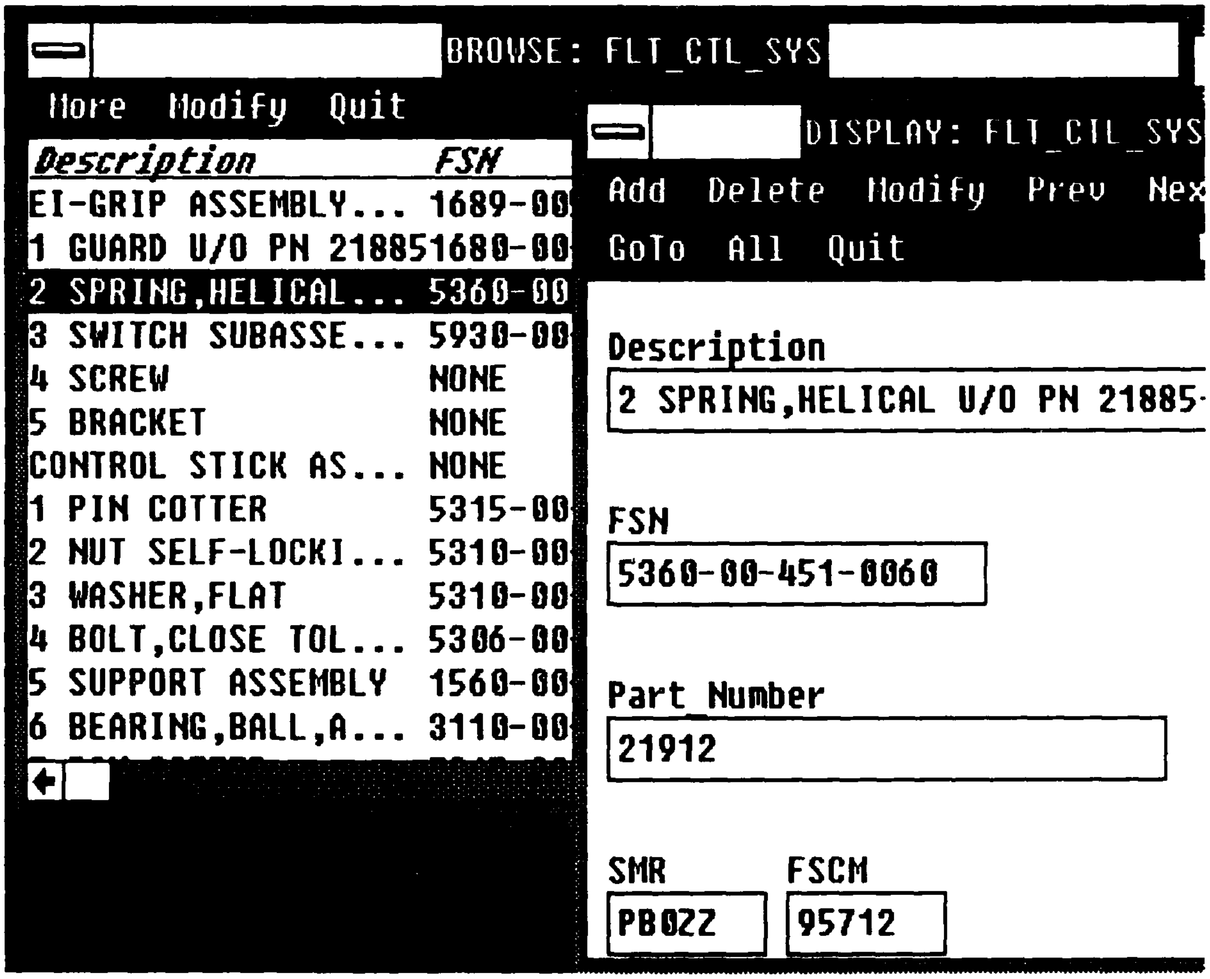
a. Menu Options

The DisplayOneWindow menu options provide the user data manipulation and browsing capabilities. The **Add** menu option displays an AddOneWindow. We see in Figure 3.9 an AddOneWindow containing a collection of edit boxes and buttons for adding instances of new data to the database. When the user quits out of this window the data will be automatically displayed in the DisplayOneWindow.



Description	FSN	Part Number	SA
EI-GRIP ASSEMBLY...	1689-00569-9573	22228	PA
1 GUARD U/O PN 21885	1680-00-450-1851	21911	PB
2 SPRING,HELICAL...	5360-00-451-0060	21912	PB
3 SWITCH SUBASSE...	5930-00-612-6993	21230	PA
4 SCREW	NONE	PM21402	XD
5 BRACKET	NONE	PM21254	XD
CONTROL STICK AS...	NONE	209-001-301-3	A0
1 PIN COTTER	5315-00-815-1405	MS24665-151	PA
2 NUT SELF-LOCKI...	5310-00-90[illegible]-9421	MS178825-5	PA
3 WASHER,FLAT	5310-00-187-2399	21230	PA
4 BOLT,CLOSE TOL...	5306-00-180-0473	PAOZZ	XD
5 SUPPORT ASSEMBLY	1560-00-917-1799	209-001-316-1	XD
6 BEARING,BALL,A...	3110-00-158-6298	DW5	PA
7 PIN,COTTER	5315-00-815-1405	MS24665-151	PA
8 NUT SELF-LOCKI...	5310-00-961-8390	MS178825-4	PA
9 NUT SELF-LOCKI...	5310-00-807-1474	MS21042L3	PA
9A WASHER,FLAT	5310-00-167-0753	AN960PD	PA
9B SCREW,MACHINE	5305-00-989-7435	MS35207-264	PA

Figure 3.6 ListMembers Window for the All at Once Menu Option



Description	FSN
EI-GRIP ASSEMBLY...	1689-00
1 GUARD U/O PN 21885	1680-00
2 SPRING,HELICAL...	5360-00
3 SWITCH SUBASSE...	5930-00
4 SCREW	NONE
5 BRACKET	NONE
CONTROL STICK AS...	NONE
1 PIN COTTER	5315-00
2 NUT SELF-LOCKI...	5310-00
3 WASHER,FLAT	5310-00
4 BOLT,CLOSE TOL...	5306-00
5 SUPPORT ASSEMBLY	1560-00
6 BEARING,BALL,A...	3110-00

Description
2 SPRING,HELICAL U/O PN 21885

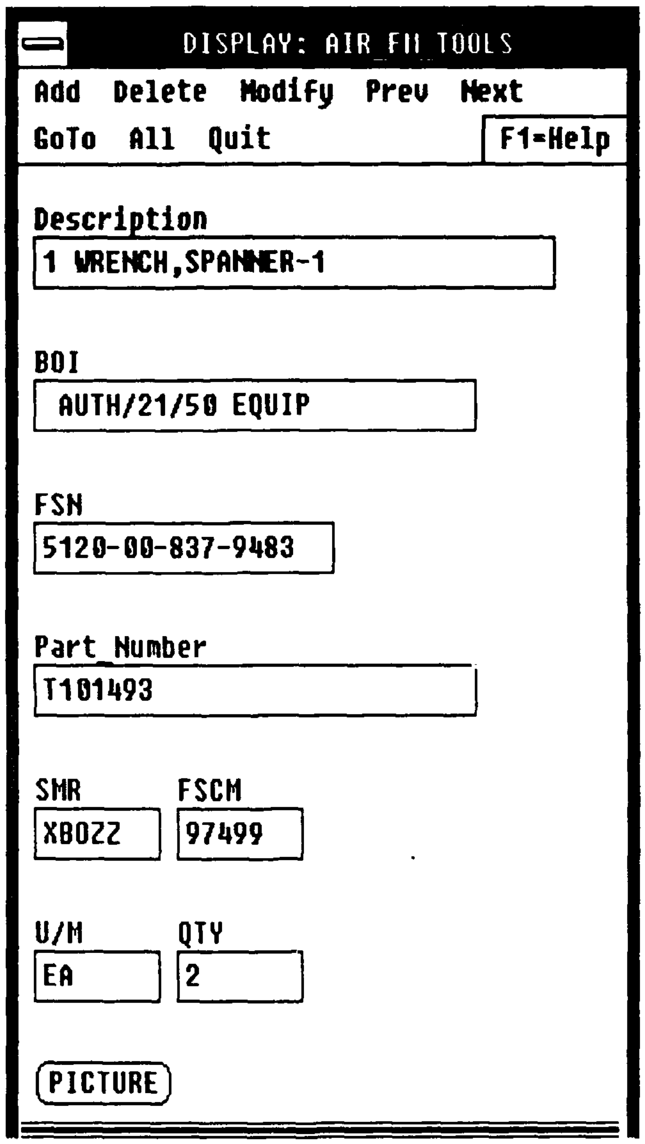
FSN
5360-00-451-0060

Part_Number
21912

SMR
PB0ZZ

FSCM
95712

Figure 3.7 DisplayOne Window from the More Menu Option


DISPLAY: AIR_FM TOOLS

Description
1 WRENCH,SPANNER-1

BOI
AUTH/21/50 EQUIP

FSN
5120-00-837-9483

Part_Number
T101493

SMR
XBOZZ

FSCM
97499

U/M
EA

QTY
2

Figure 3.8 DisplayOne Window from the One by One Menu Option

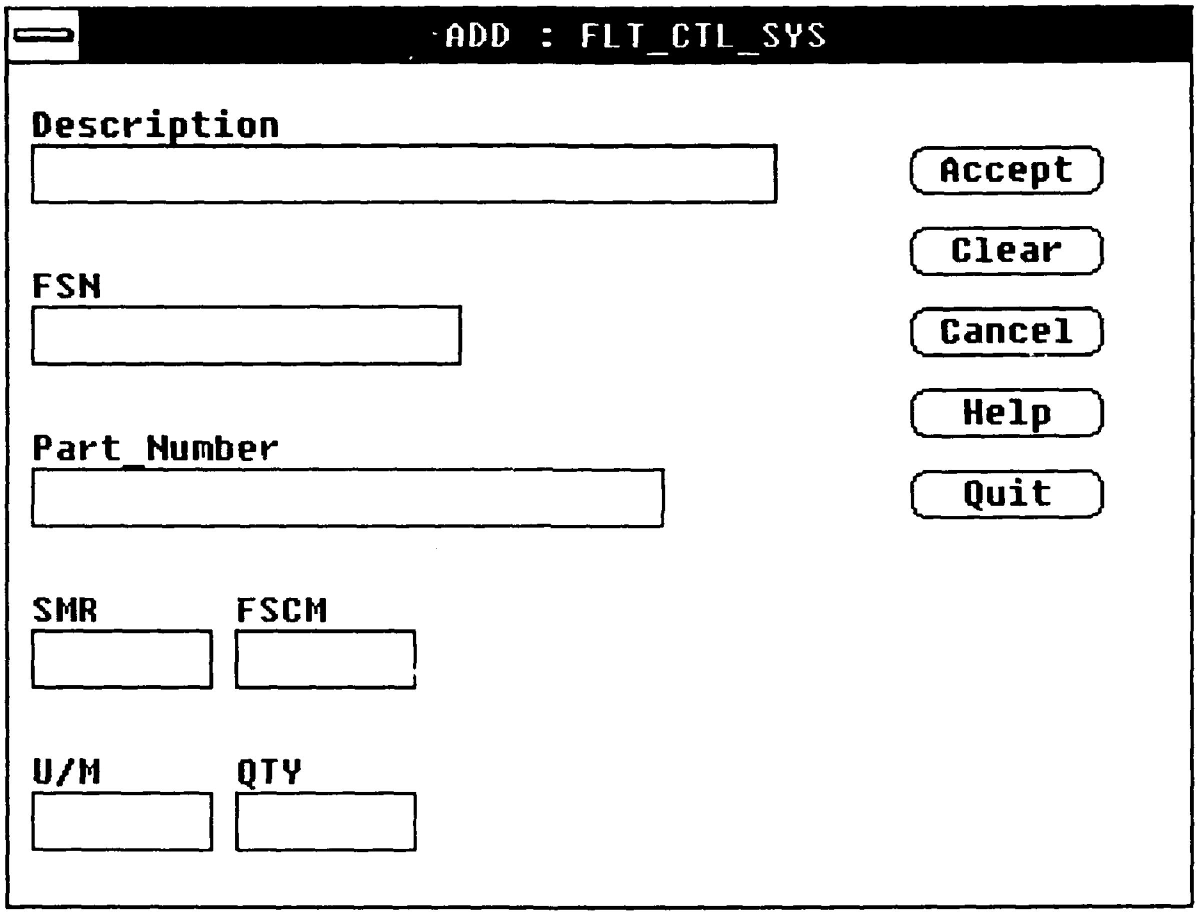


Figure 3.9 AddOne Window for the Add Menu Option

The **Delete** option has the same features as the **Add** option. A DeleteOneWindow is shown in Figure 3.10. The delete option deletes data from the database. Note that the **Add**, **Delete**, and **Modify** options are not yet fully implemented. Only the visual interfaces are presented to demonstrate the prototype.

We have full browsing facilities with the **Prev**, **Next**, **GoTo**, and **All** menu options. Selecting **Prev** or **Next** steps the user, one by one, through the 'previous' or through the 'next' tuple in the database. The **GoTo** is a pull down menu option that enables the user to go to the **First**, **Last**, or to the **Ith** tuple in the database. A separate dialogue box allows the user to fill in the Ith number they desire. The **All** allows us to view the overall database as a whole. This option is the same as the **All at Once** in the DML window, Figure 3.11.

b. Format

The remainder of the DisplayOneWindow in Figure 3.8 contains data for exactly one instance of data. As we see in Figure 3.8, there is one box for each bit of information. Above each box is the attribute associated with each bit of data. The attributes are read in from the databases' schema file, .SCH. The associated tuples for respective attributes are read in from a data file .DAT. Appendices C and D, respectively, contain the schema and data files used for the AH1S database. Editing is

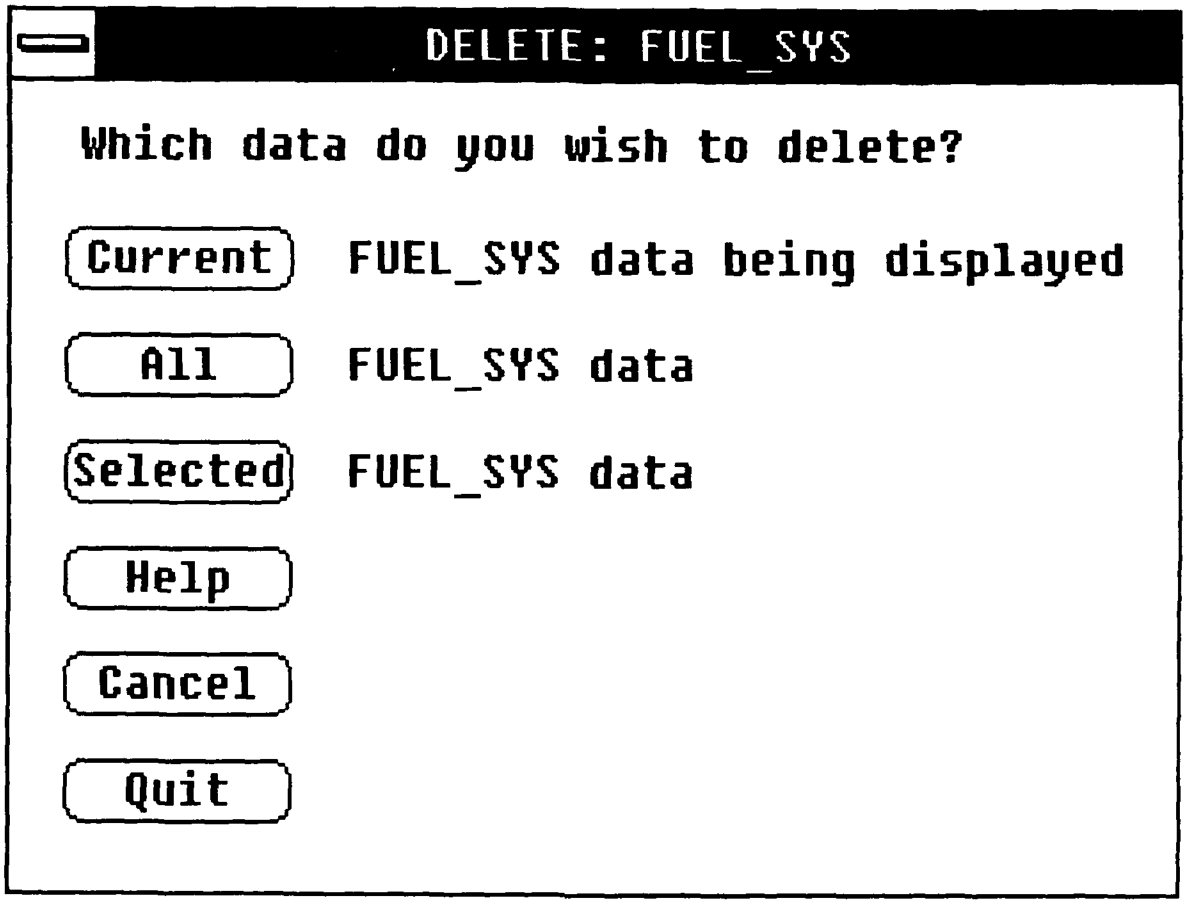
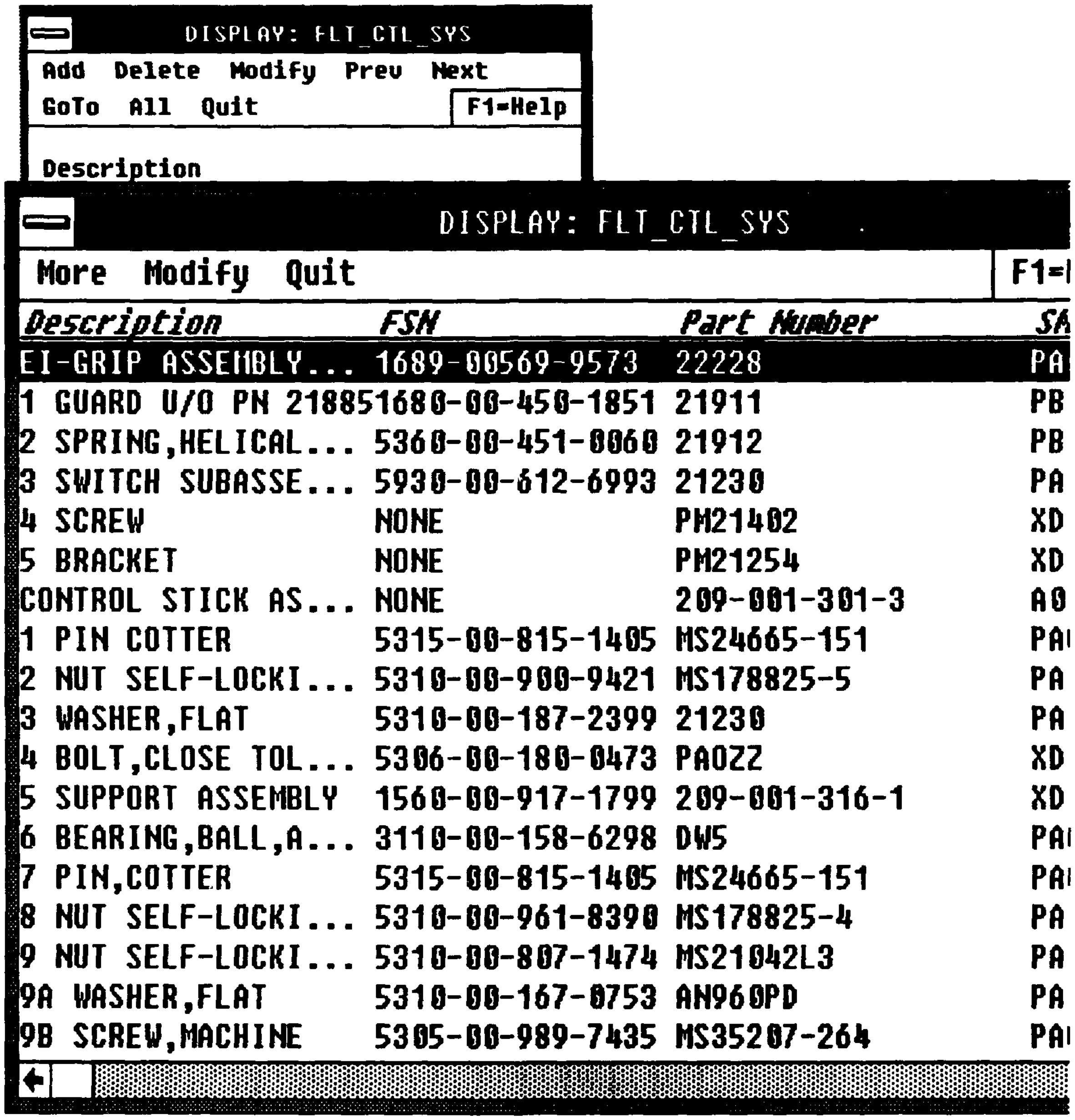


Figure 3.10 DeleteOne Window from the Delete Menu Option



```
Description          FSN               Part Number        SA
EI-GRIP ASSEMBLY...  1689-00569-9573   22228              PA
1 GUARD U/O PN 218851680-00-450-1851   21911              PB
2 SPRING,HELICAL...  5360-00-451-0060  21912              PB
3 SWITCH SUBASSE...  5930-00-612-6993  21230              PA
4 SCREW              NONE              PM21402            XD
5 BRACKET            NONE              PM21254            XD
CONTROL STICK AS...  NONE              209-001-301-3      A0
1 PIN COTTER         5315-00-815-1405  MS24665-151        PA
2 NUT SELF-LOCKI...  5310-00-900-9421  MS178825-5         PA
3 WASHER,FLAT        5310-00-187-2399  21230              PA
4 BOLT,CLOSE TOL...  5306-00-180-0473  PAOZZ              XD
5 SUPPORT ASSEMBLY   1560-00-917-1799  209-001-316-1      XD
6 BEARING,BALL,A...  3110-00-158-6298  DW5                PA
7 PIN,COTTER         5315-00-815-1405  MS24665-151        PA
8 NUT SELF-LOCKI...  5310-00-961-8390  MS178825-4         PA
9 NUT SELF-LOCKI...  5310-00-807-1474  MS21042L3          PA
9A WASHER,FLAT       5310-00-167-0753  AN960PD            PA
9B SCREW,MACHINE     5305-00-989-7435  MS35207-264        PA
```

Figure 3.11 Listmembers Window for the All Menu Option

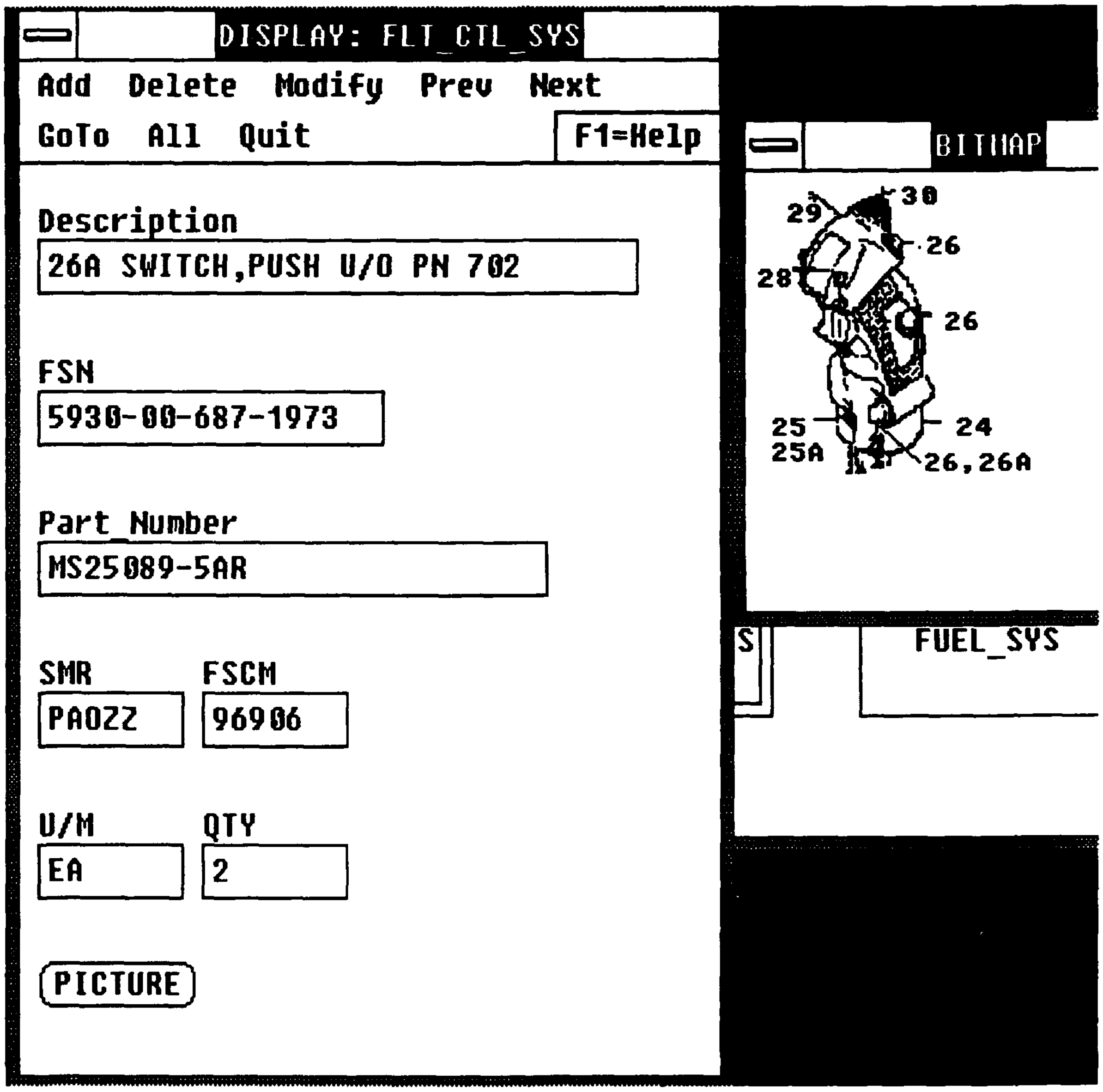
prohibited inside the data boxes. Any attempt to do so will generate an error box.

The database designer customizes the presentation of data boxes and the size of the DisplayOneWindow. The amount of data to be displayed determines the size of the boxes and the window. A Template file .TPL stores display format information for each database created. See Appendix E for the template file used to create the Display window in Figure 3.8.

If you notice in Figure 3.8, there is a button at the bottom that indicates "Picture." Each database schema has the option of displaying a bitmap image.

4. **Bitmap window**

Normally, it is infrequent that databases offer a graphics display capability as an attribute to its schema. GLAD offers us that capability as shown in Figure 3.8. The user clicks once on the button labeled "Picture" and the bitmap stored in that instance of data in the current DisplayOneWindow is displayed in a "Bitmap Window." Figures 3.12, 3.13 and 3.14 show typical photos displayed from the attack helicopter database. The bitmap window is movable and sizable so that it can be continually displayed while working on data in the DisplayOneWindow. This is an important feature. The first number in the nomenclature field of the Display Window corresponds to a component on the picture in the Bitmap Window. As the user scrolls



DISPLAY: FLT_CTL_SYS

Add Delete Modify Prev Next

GoTo All Quit F1=Help

Description
26A SWITCH,PUSH U/O PN 702

FSN
5930-00-687-1973

Part_Number
MS25089-5AR

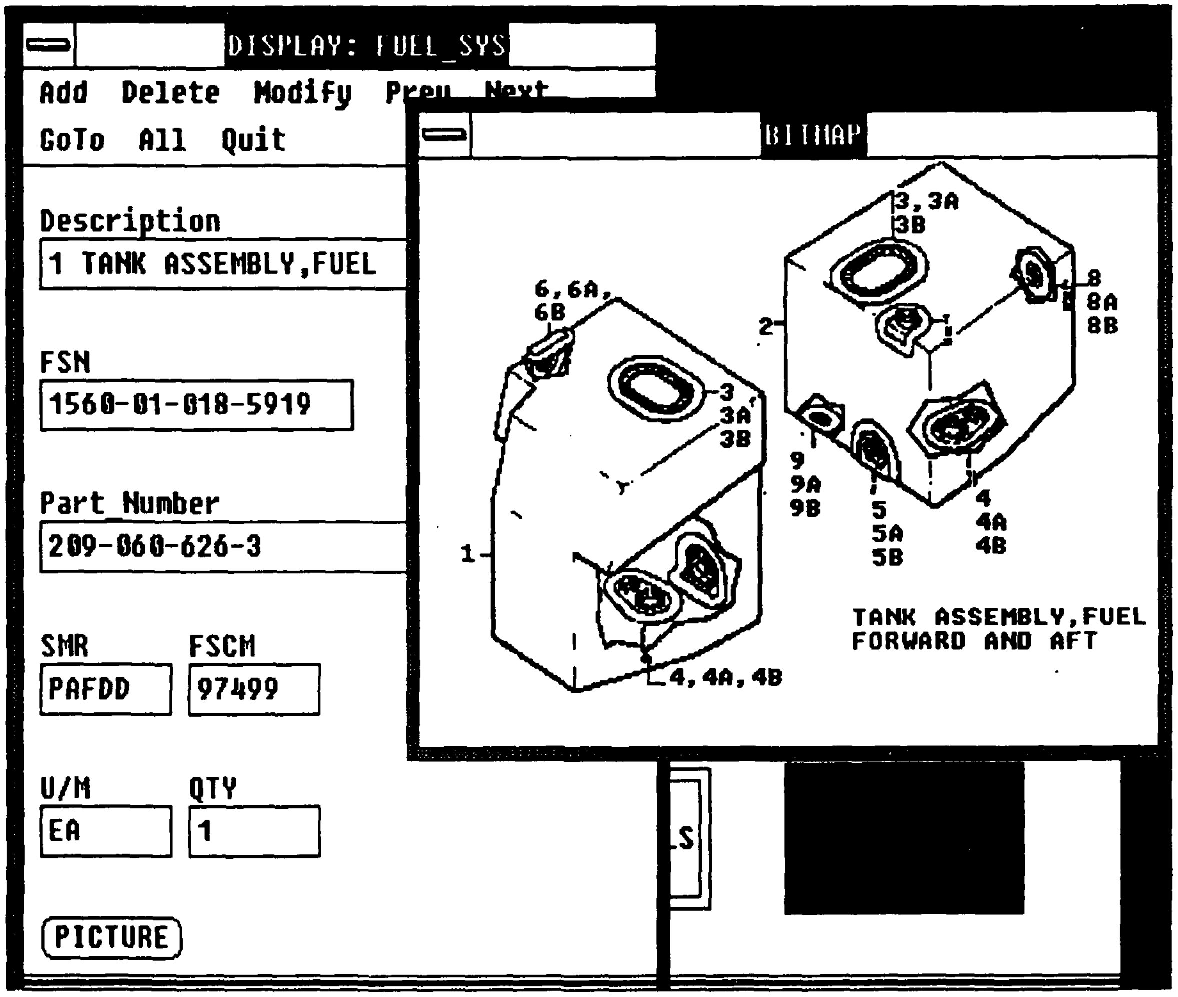
SMR
PAOZZ

FSCM
96906

U/M
EA

QTY
2

Figure 3.12 Bitmap Window selected with Picture Button



DISPLAY: FUEL_SYS

Add Delete Modify Prev Next
GoTo All Quit

Description
1 TANK ASSEMBLY,FUEL

FSN
1560-01-018-5919

Part Number
209-060-626-3

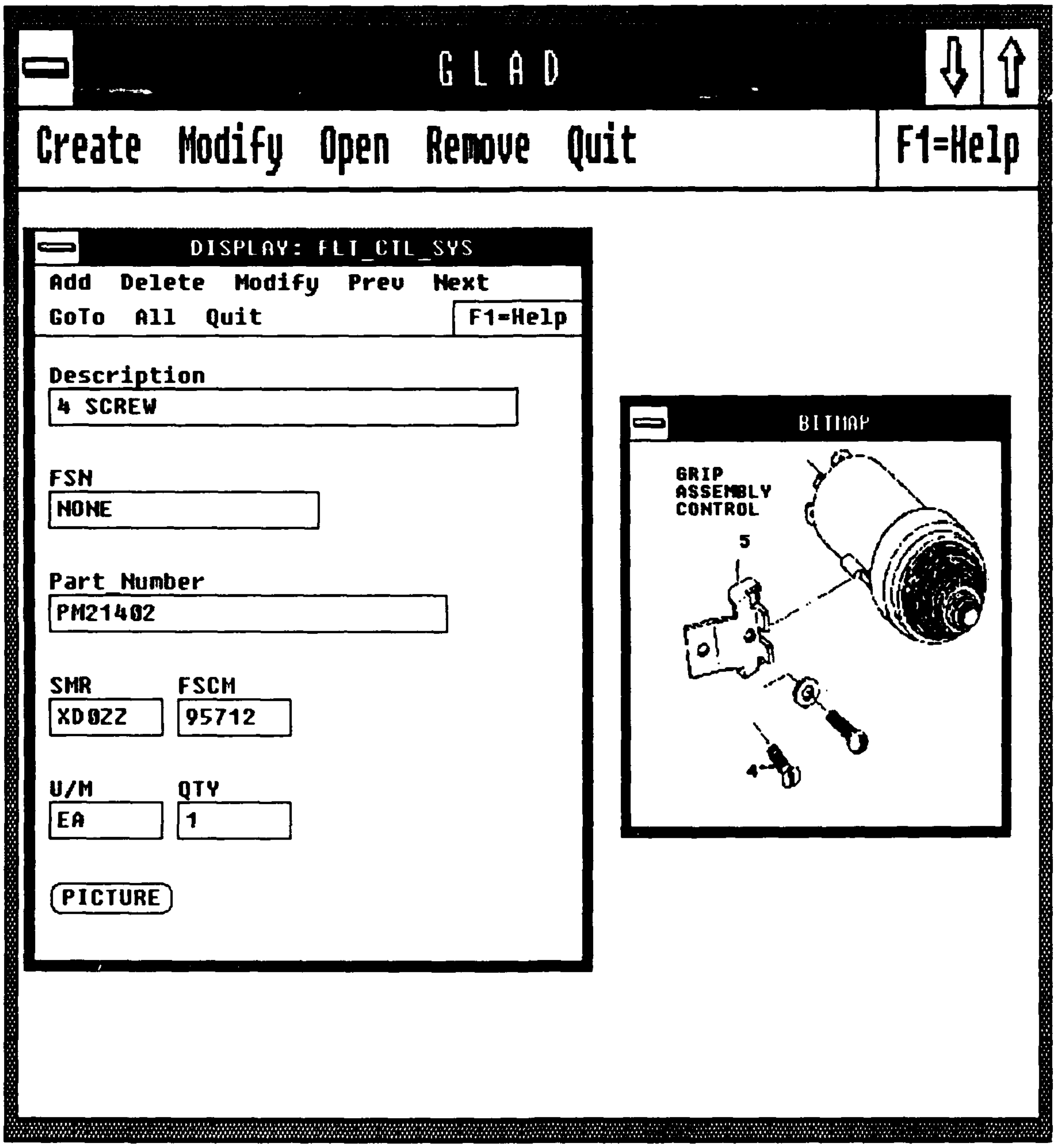
SMR	FSCM
PAFDD	97499

U/M	QTY
EA	1

PICTURE

BITMAP

Figure 3.13 Bitmap Window Selected with Picture Button



Description
4 SCREW

FSN
NONE

Part_Number
PM21402

SMR	FSCM
XD0ZZ	95712

U/M	QTY
EA	1

Figure 3.14 Bitmap Window Selected with Picture Button

through the database using the various menu options, he or she matches up the part with that shown in the picture. This procedure is normally performed manually by maintenance and logistical personnel for verification and quality control purposes with bulky technical manuals.

There can be more than one bitmap stored in the schema at a time. This feature is illustrated by the multiple buttons displayed in Figure 3.15. At this point only one bitmap window at a time is permitted to be displayed because of memory limitations. If a user attempts to open more than one picture at a time, a dialogue error box will be displayed with a warning message to close the previous Bitmaps window, Figure 3.16.

The Bitmap Window can be closed by either double clicking with the mouse on the system box in the upper left hand of the window or selecting **Close** from the window's pull down menu box. Sizing is accomplished by 'grabbing' the border of the window and resizing or selecting **Size** option from the window's pull down menu.

Before we leave this section, notice back in Figure 3.8 the object called "Cobra." This object is not an entity set from the point that it contains attributes and tuples of data. The Cobra object illustrates the flexibility found in programming GLAD objects. Upon selection of the Cobra object and the **One by one** menu option, a normal DisplayOneWindow window is displayed. The only difference is that a number of picture buttons are

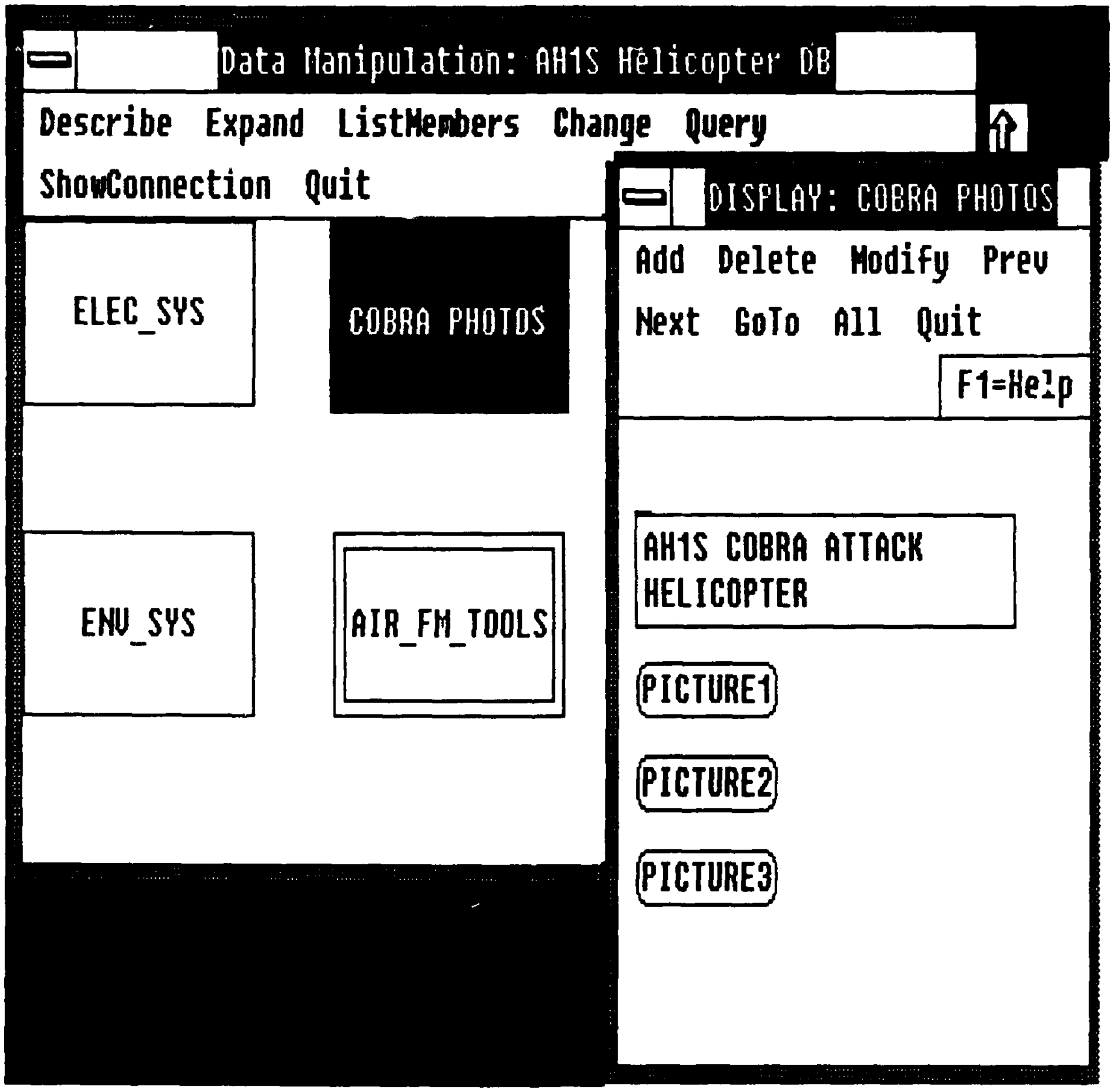
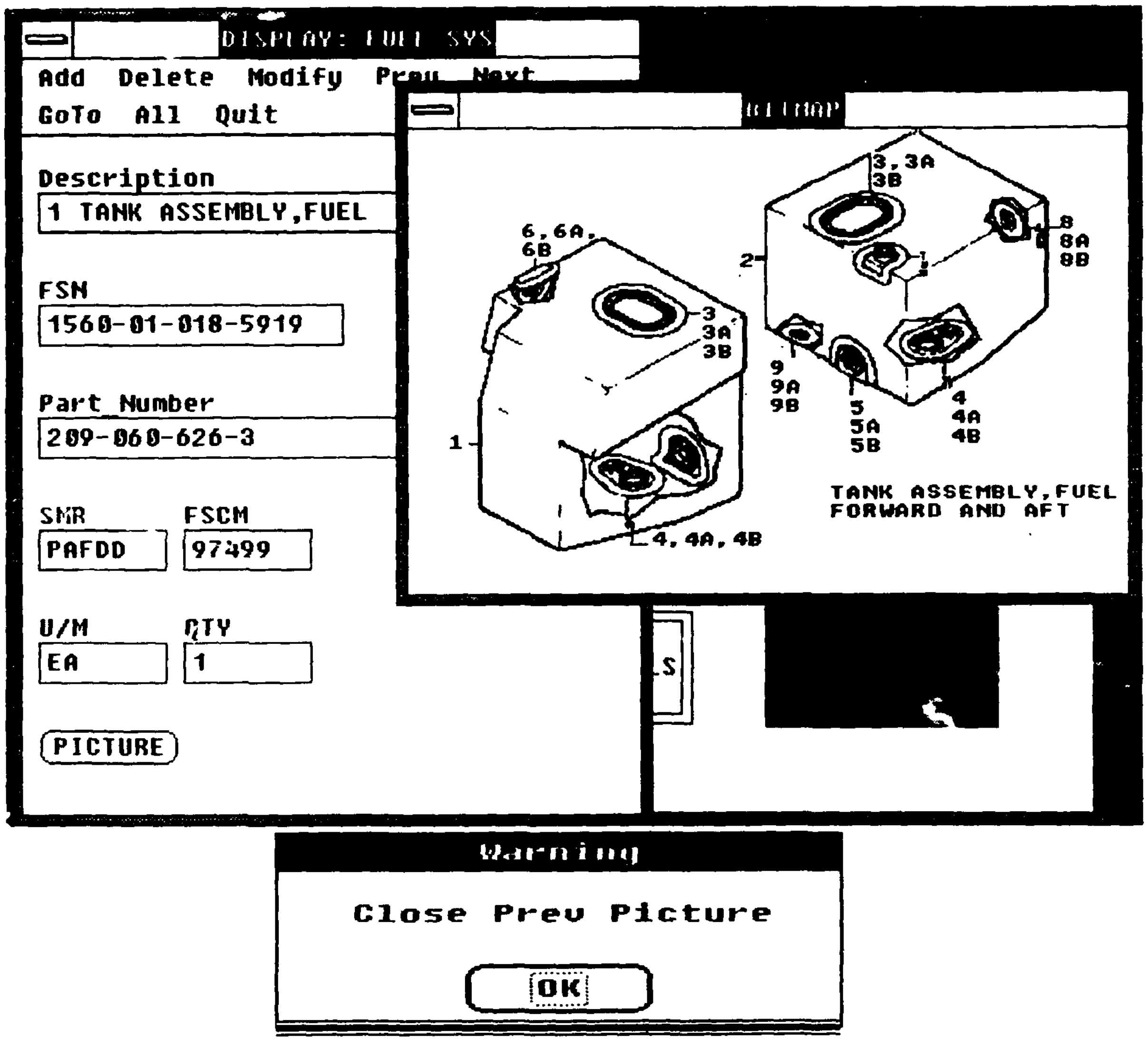


Figure 3.15 Multiple Bitmap Buttons



DISPLAY: FUEL SYS

Add Delete Modify Prev Next
GoTo All Quit

Description
1 TANK ASSEMBLY,FUEL

FSN
1560-01-018-5919

Part Number
209-060-626-3

SMR
PAFDD

FSCM
97499

U/M
EA

QTY
1

PICTURE

BITMAP

Warning

Close Prev Picture

OK

Figure 3.16 Bitmap Error Dialogue Box

presented and no data, Figure 3.15. The Bitmap windows here are various composite shots of the cobra helicopter for orientations purposes. The multiple buttons represent different views. For example, by pressing the first button a color frontal view of a Cobra helicopter will be displayed, Figure 3.17. By pressing the second button, the Cobra's engine department will be displayed, Figure 3.18. Finally, the third button displays the Cobra's frame, Figure 3.19. GLAD allows flexibility in the design of the system with minimal changes.

C. BITMAPS

1. **Implementation**

As mentioned earlier, the difficult part of this research was displaying bitmaps inside of GLAD. The object-oriented aspects of programming in Actor facilitated the successful implementation of this goal. Windows are objects. Instances of data in the database are objects. Classes and methods are objects. Bitmaps are objects. Everything is an object. Once this basic concept was understood, programming in Actor was relatively simple.

The bitmap display implementation involved scanning pictures, creating a separate "Bitmaps" class and the modification of other classes within GLAD. Since the modifications to other GLAD classes involved simply the creation and display of buttons; only a discussion of scanning and the implementation of the Bitmap class will be

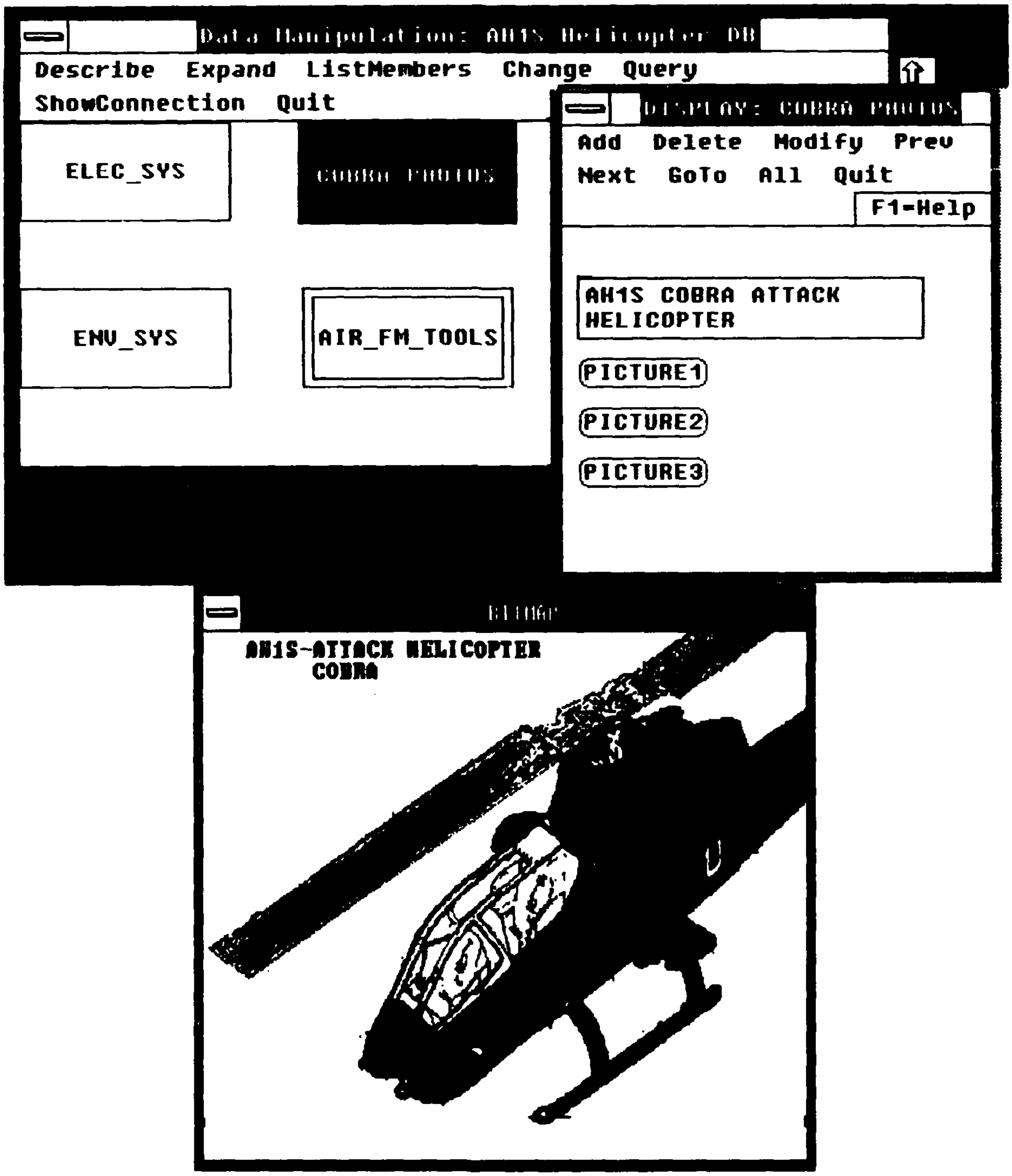
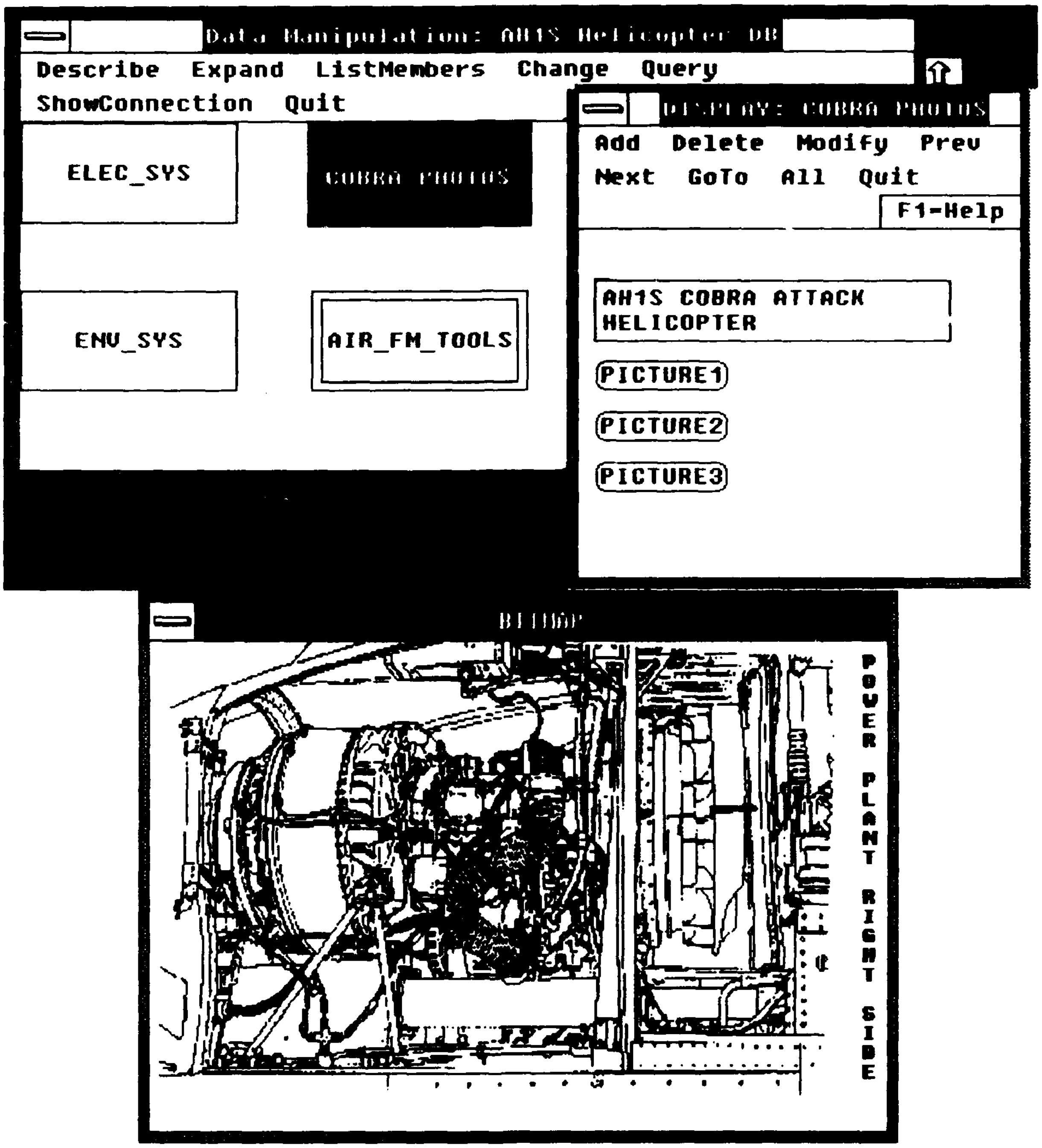


Figure 3.17 Bitmap from Selecting Button One



AH1S COBRA ATTACK HELICOPTER

Figure 3.18 Bitmap from Selecting Button Two

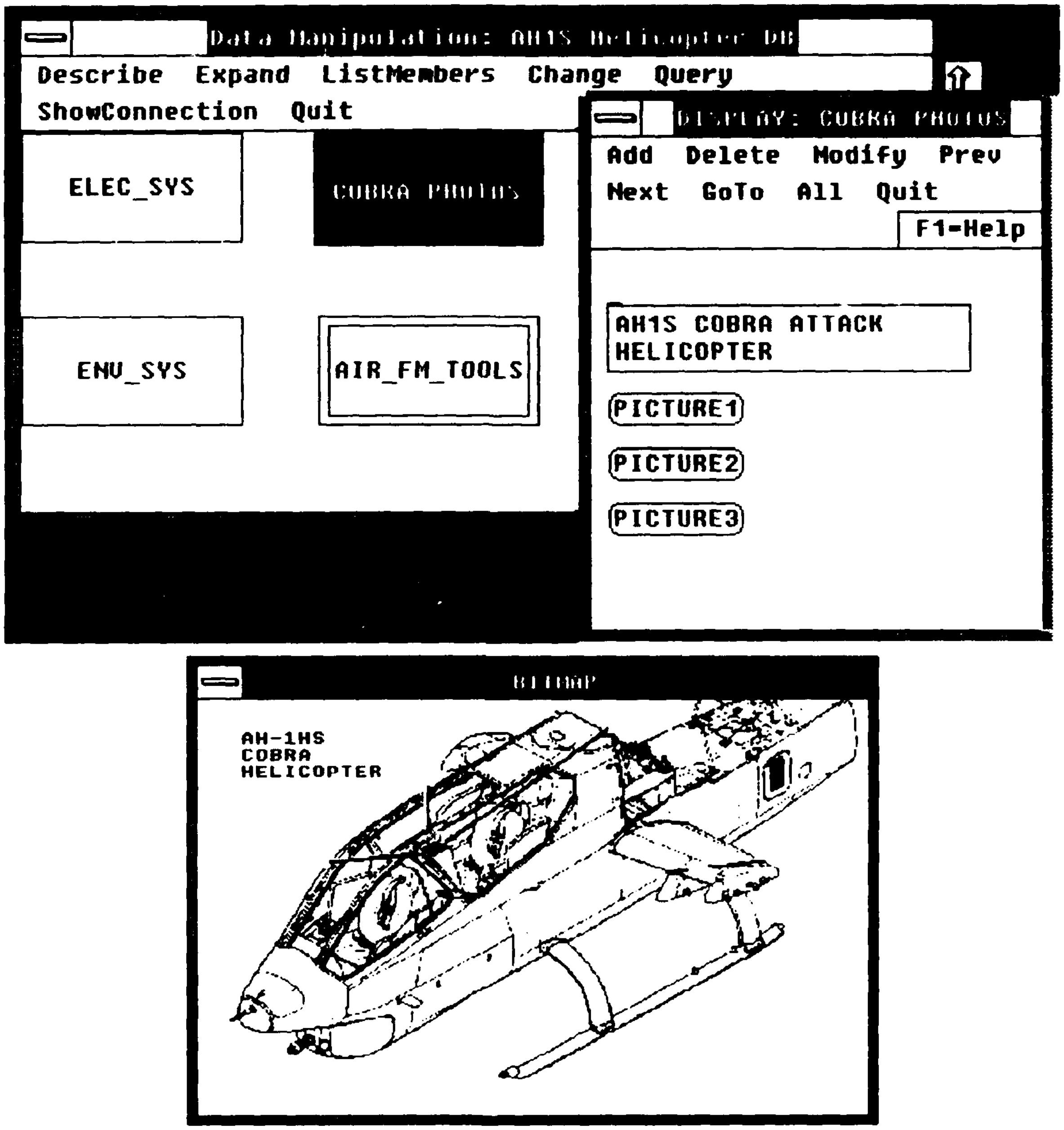


Figure 3.19 Bitmap from Selecting Button Three

covered in the following sections. The code can be found in Appendix A.

a. Scanning of pictures

All pictures for this research were scanned into graphic files on a Hewlett Packard ScanJet Scanner. The pictures were standard black and white. Large pictures were reduced to 64K bytes or lower. Initial graphic files can have any of the commercial paint program extensions, however the final one must be in Windows Paint's .MSP format. The quality of the scanner was such that many of the original pictures required editing of text and lines. Editing for these pictures were accomplished inside MS-Windows Paint program. One of the problems that we found with MS-Windows Paint is that 90 degree rotations of pictures were not possible. This presented some problems because a reasonable orientation could be scanned to fit MS-Windows Paint program for a number of pictures. Subsequently, some pictures were scanned in and edited in Microsoft's Paintbrush paint program. The Paintbrush files were then converted to MS-Window's Paint format. A public domain graphics conversion software package was used to convert files. However, any file conversion utility, found in most paint and scanner programs, could have been used.

All final products were converted to MS-Windows Paint .MSP files. These files were then loaded from MS-Windows Paint into the MS-Windows Clipboard. They were then downloaded into a MS-Windows program called Clipfile and saved as bitmap files. Clipfile is a MS-Windows utility program that saves graphic files downloaded from the Clipboard as bitmap graphic files with .BMP extensions. Clipboard also displays bitmap structure information, such as height, width, bits per pixel and so on. This information was critical in capturing bitmap definitions for the Bitmaps class method. Final bitmaps were then stored onto hard disk for use in the databases.

b. Bitmaps Class

The Bitmap class is a descendant of the DisplayOneWindow class. This design decision was based on the fact that pictures were likely to be found in **One by one** instances only. The Bitmap class has been written in the true sense of optimizing modularity and reusability. This class is general enough that it can be a descendant of just about any class in the GLAD system.

The code at Appendix A lists all the methods used to display the bitmaps. However, we will briefly go through the general sequence of events. As we know, messages are sent to objects who perform some operations on themselves, however, an object has to be created first. A message is sent to an object to create itself. The DisplayOneWindow class sends a message to the Bitmaps class to create an instance of itself when the picture button is pushed in the Display Window, Figure 3.8. We called the instance of Bitmaps class "picture." Once an instance of the Bitmap class is created, DisplayOneWindow class sends a message to it to show the picture (showthepic method) of the currently selected object in the Display window. This message starts the ball rolling. One of the arguments in the message to "picture" is the string name of the externally stored bitmap file to be displayed. Where are these string names stored? They are stored as data in the data files, .DAT, of as part of the BITMAPS attribute field. Currently, the last attribute(s) of the schema and the tuples is a bitmap field.

Upon receipt of the showthePic message, Bitmaps opens the bitmap graphics file and reads in header information. The bitmap header information includes: height and width of the bitmap in bits, bits per pixel, number of bit planes and width of the bitmap bytes. This information is necessary to lock down enough memory to read in the actual bitmap bits and to perform window sizing calculations. Once the actual bitmap bits are read into memory, the file is closed. A Bitmap window is then created, moved to a predetermined location and "painted" with the contents of the bitmap bits waiting out in memory. The window is then displayed as in Figure 3.12.

2. Memory and Storage

One of the significant implementation difficulties or concerns was that of memory management and availability. Bitmaps inherently requires large amounts of memory. The average size of the bitmaps used for this research were 20k bytes. These seem to be respectable size bitmaps when displayed. Larger bitmaps up to 64k bytes are allowed to be displayed. During testing, it was determined that multiple bitmaps could not successfully be displayed on machines with only 1MB of Random Access Memory (RAM). We determined that Windows, Actor and GLAD require large amounts of unexplained RAM. The MS-Windows manager does a fair job of freeing up memory when it can, but bitmaps require amounts of memory too large to accomodate at once. The final implementation design decision called for the display of only one bitmap at a time. The single bitmap display decision proved correct because it has been run successfully on a number of different machines. An easy modification is available, for machines with greater than 1MB, to display multiple bitmaps. The comments in the "showmemberPict" method of the DisplayOneWindow class in Appendix A tells how it can be accomplished.

GLAD can also be configured in "stand-alone" application, which significantly reduces memory requirements. The current version (GLADV02) has been successfully implemented as a stand-alone application. Once you have developed the classes and methods needed to run your application, Actor outlines procedures for developing a stand-alone application.

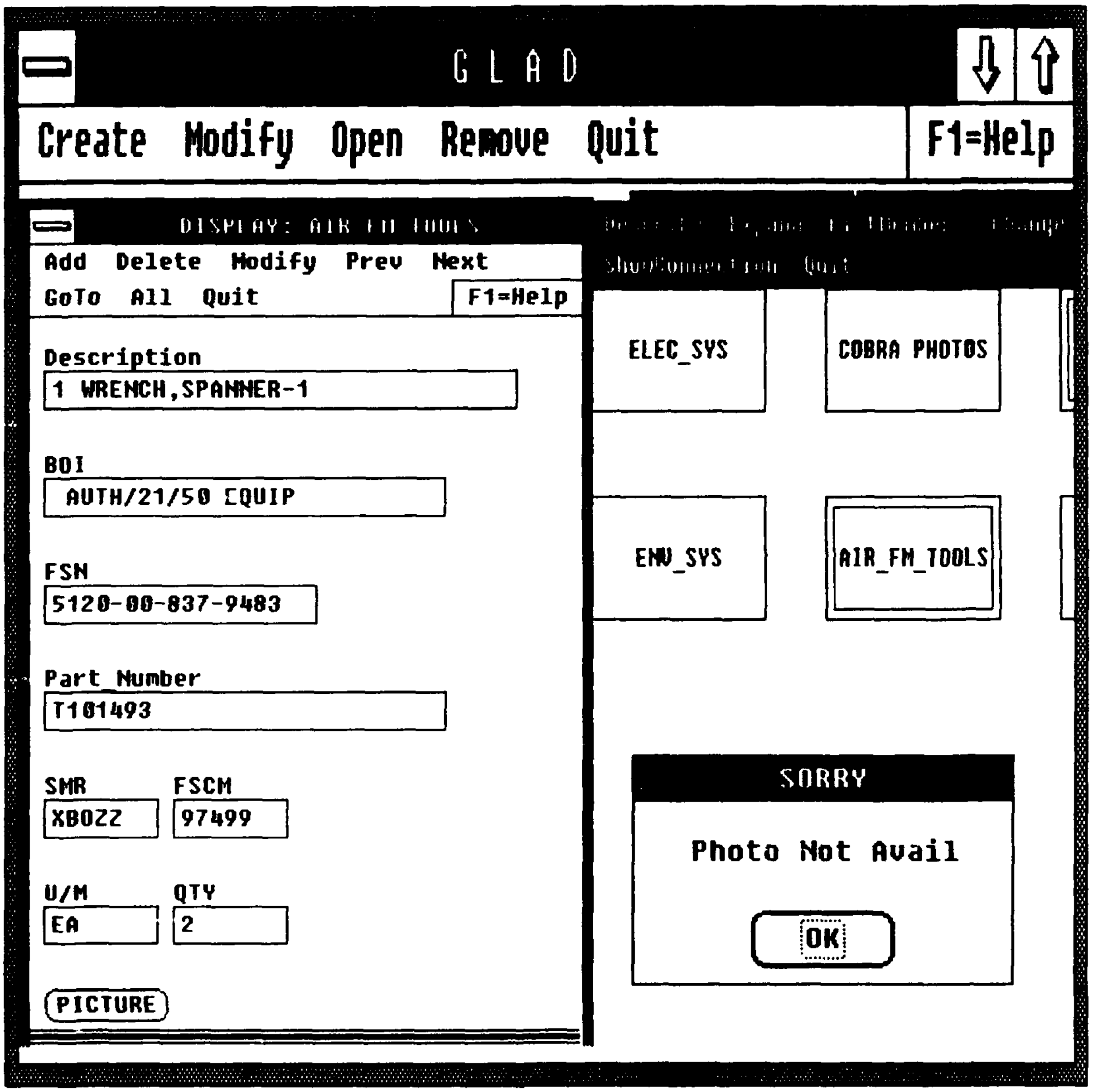
The two pieces of any Actor application are the kernel (.EXE file) and the image (.IMA file). In our application, GLADV02.EXE files is the actual executable file which manages the execution of code and contains

Actor's machine language primitives. The GLADVO2.IMA file contains the compiled code that utilizes the built in primitives in the .EXE file. The whole idea in developing a stand-alone is to exclude all unnecessary classes and methods, thus reducing memory requirements. Methods found in the debuggers, browsers and so on, are not needed in the final application.

Actor has a SMALL.IMA file that lets the user start with a minimum system. We created a load file containing all of our classes and those required Actor classes not included in the SMALL.IMA, to use as input to SMALL.IMA, see Appendix F. We then loaded and compiled this file on top of SMALL.IMA to produce the smaller GLADVO2.IMA. Since the .IMA is an Actor unique file and unfamiliar to non-Actor literate users, we generated an executable GLADVO2 file. This involved resource compiling our GLAD unique resources and constant definitions, along with those needed by ACTOR, into a GLADVO2.EXE file. Appendix F contains the files used to produce GLADVO2.EXE. Subsequently, users can start GLAD by using the .EXE file.

The memory savings generated by a stand-alone allows multiple bitmaps to be displayed up to point. Given the dynamic nature of the windowing environment, more than two bitmap windows on a 1MB machine still produced memory errors on subsequent testing. In our opinion, the one bitmap display limit implementation is the best solution for 1MB machines. However, the stand-alone configuration frees up substantial memory for other application uses.

All bitmaps are currently stored in separate directories on a hard disk and must be available at run time. If they are not, a photo not available error message is displayed in an error box, Figure 3.20. Storage considerations will have to be examined for applications with a large number of bitmaps. The technology exists in the form of CD ROM and other external storage devices. The size of typical repair parts manual, with a little discipline, should fit in 2MB of hard disk without difficulty.



Description: 1 WRENCH,SPANNER-1

BOI: AUTH/21/50 EQUIP

FSN: 5120-00-837-9483

Part_Number: T101493

SMR: XBOZZ

FSCM: 97499

U/M: EA

QTY: 2

SORRY

Photo Not Avail

Figure 3.20 Bitmap Photo Not Available Dialogue Box

IV. CONCLUSIONS

A. FUTURE ENHANCEMENTS

There are a multitude of enhancements that could be applied as extensions to this research. We will list a few:

- Multiple bitmaps displayed. As we discussed earlier, this is easily achievable. The only consideration is the memory. Even this may not be an obstacle in the near future with the drop in RAM costs.

- Color bitmaps. Color bitmaps were implemented by scanning in a picture and painting it in one of the various paint programs. This may require a significant amount of work for a large number of pictures. A color scanner would enhance the quality of the bitmaps.

- Iconic bitmaps. This may not be necessary with only one bitmap displayed. If multiple bitmaps, then we may consider this feature to unclutter the screen while performing operations on the database.

- Print option. A printer class with a "print" menu option to output the bitmaps, data files, or instances of data from the Display window would provide a nice interface.

- Automatic requisition generation. This may be little difficult because the scope of the problem is not just limited to one instance of data of a selected object. We envision in the ULLS environment of automatically generating and printing a repair parts requisitions from the data in the Display window. Additionally, all appropriate historical documents and related management records would be simultaneously updated. We think that the GLAD interface as it stands would suffice with the addition of a print class. Only the logic would need to be written.

- Telecommunications interface. An interesting idea. The above enhancements are achievable. Technology is advancing to the point that some of the memory related hindrances may soon disappear and allow us to proceed forward with the display of unlimited windows and bitmaps.

B. DISCUSSION AND BENEFITS OF THE RESEARCH

The U.S. Army is moving forward in the field of automation. As they move forward, we feel that the human-computer interface issue should become a key factor in system design and acquisition. Specifically, in the area of databases, we need to take the opportunity to examine the visual interfaces that are being explored in the research community. We attempted to do exactly that with GLAD.

Even though we did not produce a fully implemented prototype, we illustrated the potential of providing a visual interface to maintenance and logistics database systems. More importantly, which was the substance of this research, graphics display within databases is attainable. Database systems that run on microcomputers, such as ULLS, are good candidates for GLAD type interfaces.

Other applications besides those in the military could benefit from the findings of this research project. For example, we tested a simple university database and determined the a desirable enhancement would the bitmap display of photos of students, faculty and staff. The photos are stored as bitmaps right along with other data in the tuples. Applications of this type are limitless.

GLAD now has the capability to display graphics, which moves it another step closer towards becoming a viable visual interface to databases. The trend is to move away from the bulky paper environments and to 'paperless' ships, motorpools, maintenance facilities and supply depots. The work completed in this thesis is a positive step in that direction.

APPENDIX A. SOURCE CODE LISTING

The listing that follows includes those classes modified or created for the implementation of this thesis.

1. AddOneWindow Class (Modified)

```
/* a window class to add new instances of GLAD objects. */!!

inherit(OneMemWindow, #AddOneWindow, #(addedMembers /*ordered
collection of                 newly added instances*/
buttons      /*array of buttons*/), 2, nil)!!

now(AddOneWindowClass)!!

now(AddOneWindow)!!

/*move the focus to the next edit control*/
Def nextEC(self,idx)
{
  idx := (idx+1) mod size(editControl);
  setFocus(editControl[idx])
}!!

Def errorCheck(self,fields)
{
  /*do the input error checking*/
  ^0
}!!

Def create(self,par,wName,rect,style)
{
  ^create(self:Window,par,wName,rect,
          WS_CAPTION bitOr WS_SYSMENU bitOr WS_POPUP)
}!!

Def command(self,wp,lp)
{
  /*handles only the buttons; ignore edit controls*/
  if high(lp) <> 1 and menuID[wp]
    perform(self,menuID[wp])
  endif
}!!

Def help(selflaStr)
{aStr :=asciiz("Add One Window");
```

```
  pcall(Lib.procs[#GUIDANCESETCONTEXT],HGuide,
  lP(aStr),1);
  freeHandle(aStr);
} !!

Def cancel(self)
{
}!!

Def displayButtons(self | left, top)
{
  left := right(clientRect(self)) - 100;
  top  :=   top(clientRect(self)) +  30;
  do(over(0,5), {using(idx)

setCRect(buttons[idx],rect(left,top,left+70,top+20));
                  moveWindow(buttons[idx]);
                  show(buttons[idx],1);
                  top := top + 30 })
}!!

Def createButtons(self)
{
  buttons := new(Array,5);
  buttons[0] := newPush(Button,100,self,"Accept");
  buttons[1] := newPush(Button,101,self,"Clear");
  buttons[2] := newPush(Button,102,self,"Cancel");
  buttons[3] := newPush(Button,103,self,"Help");
  buttons[4] := newPush(Button,104,self,"Quit");
}!!

Def paint(self,hdc)
{
  paint(self:OneMemWindow,hdc);
  displayButtons(self)
}!!

Def expandWindow(self | wRect, left, top, right, bottom,
lastEC,bmCntminus)
{
  wRect:= windowRect(self);
  left:=(x(screenSize()) -  width(wRect))/2;
  top :=(y(screenSize()) - height(wRect))/2;
  right := left + width(wRect) + 100;
  if bmCnt
    bmCntminus:= bmCnt + 1;
    lastEC := template[size(attributes(selObject))-
bmCntminus]
  else
    lastEC := template[size(attributes(selObject))-1];
  endif;
  bottom := top+(lastEC[EC_LOC].y+lastEC[EC_DIM].y+2)*tmHeight

             + Call GetSystemMetrics(31 /*height of title
bar*/);
  setCRect(self,rect(left,top,right,bottom));
  moveWindow(self)
}!!

Def add(self | fields, fieldCnt, notEmpty)
{
  /*scan through the edit controls*/
  if bmCnt
    fieldCnt :=(size(attributes(selObject))-bmCnt);
  else
    fieldCnt :=size(attributes(selObject));
  endif;
  fields  := new(Array,fieldCnt);
  do(over(0,fieldCnt), {using(idx)
                        fields[idx] :=
getText(editControl[idx]);
                        notEmpty := notEmpty cor fields[idx]
<> ""});
  if notEmpty
    errorCheck(self,fields);
    add(addedMembers,fields); beep(); beep();
    setText(self,"ADD : "+name(selObject)+" ["+
                 asString(size(addedMembers))+" added]");
    clear(self);

    /*send this fields to DisplayOneWindow if there is one*/
    if class(parent) = DisplayOneWindow
      append(parent,fields)
    endif
  else
    beep();
    errorBox("NO VALUES","Type in the values and then press
Accept")
  endif
}!!

Def start(self,obj)
{
  addedMembers := new(OrderedCollection,5);
  selObject := obj;
  initTextMetrics(self);
  getbmCnt(self);
  loadTemplate(self);
  createECs(self);
  expandWindow(self);
  createButtons(self);
  setScrollRanges(self);
  show(self,1)
}!!

Def initMenuID(self)
{
  /*this is actually a button IDs since this window has no
menu*/
  menuID := %Dictionary(
                          100->#add
                          101->#clear
                          102->#cancel
                          950->#help
                          104->#close
                        )
}!!

/*clears the content of every edit controls*/
Def clear(self)
{
  do(editControl, {using(ec) setText(ec,"")})
}!!
```

2. AttribDialog Class (Modified)

```
/* class comment */!!

inherit(Dialog, #AttribDialog, #(attrList  /* List of
attributes in            the same format as
selObj.attributes */
selAttr  /* Attribute selected from                   listbox
*/
objName /* Name of the object for which these attributes apply
*/
typeList /* List of possible types of attributes including
system and user defined attributes */
typeDialog /* Type dialog box */
tempArray /* Temporary attribute array*/
obj /*Array of two elements for setting up tempArray */), 2,
nil)!!

now(AttribDialogClass)!!

/* comment */
Def  new(self,selObj | theDlg)
{
   theDlg := new(self:Behavior);
   theDlg.objName := selObj.name;
   theDlg.selAttr := new(String, 15);
   if selObj.attributes
     theDlg.attrList := selObj.attributes
   else
     theDlg.attrList := new(OrderedCollection, 15)
   endif;
   ^theDlg

}    !!

now(AttribDialog)!!

/* Searches type list for valid type names */
Def  isTypeDef(self, aColl, newName)
{
   do(aColl,
      {using(elem) if elem[0] = newName
       ^elem[1]
       endif;});
   ^nil
}     !!

Def insertString(self,aStr | ans, insertLoc)
{
  if selAttr
    insertLoc := selAttr
  else
    insertLoc := -1
  endif;
  ans := Call SendDlgItemMessage(hWnd, ATTR_LIST,
                                 LB_INSERTSTRING,insertLoc,
                                 lP(aStr));
  freeHandle(aStr);
  ^ans
}      !!

Def addString(self,aStr | ans)
{
  ans := Call SendDlgItemMessage(hWnd, ATTR_LIST,
                                  LB_ADDSTRING,0,lP(aStr));
  freeHandle(aStr);
  ^ans
}   !!

/* This method adds a new attribute to the selected objects

   attribute list */
Def addAttr(self)
{  tempArray := new(Array, 4);
   tempArray[0] := getItemText(self, ATTR_NAME);
   tempArray[1] := getItemText(self, ATTR_TYPE);
   tempArray[2] := getItemText(self, ATTR_LENGTH);
   if (tempArray[0] = "" or tempArray[1] = "" or
       tempArray[2] = "")
     errorBox("ERROR!", "Fill in all attribute fields.")
   else
     if isTypeDef(self, attrList, tempArray[0])
       errorBcx("WARNING!", "Attribute already exists.")
     else
       if tempArray[3] := isTypeDef(self, typeList,
tempArray[1])
         selAttr := insertString(self, subString(tempArray[0]
+ "          ", 0, 10)
            + ": " + tempArray[1] + "[" + tempArray[2] +
"]");
         insert(attrList, tempArray, selAttr);
         setCurSel(self, selAttr)
       else
         errorBox("WARNING!", "Invalid Attribute Type.")
       endif
     endif
   endif
}  !!

/* Clears the text edit boxes of the dialog. */
Def  initEditBox(self)
{
  setItemText(self, ATTR_NAME,"");
  setItemText(self, ATTR_TYPE, "");
  setItemText(self, ATTR_LENGTH, "");
  selAttr := nil
}   !!

/* This method deletes attributes from an object's
   attribute list */
```

```
Def  deleteAttr(self)
{
  if selAttr
    remove(attrList, selAttr);
    Call SendDlgItemMessage (hWnd,ATTR_LIST, LB_DELETESTRING,
                              selAttr,0);
    initEditBox(self);
    setCurSel(self, -1);
    selAttr := nil
  else
    errorBox("WARNING!", "Select Attribute to delete.")
  endif
}      !!

/* Selects attribute in list box and initializes the
  variables and edit boxes */
Def  selItem(self)
{
  selAttr := Call SendDlgItemMessage (hWnd,ATTR_LIST,
            LB_GETCURSEL,0,0);
  if (selAttr >= 0 and selAttr < size(attrList))
    setItemText(self, ATTR_NAME,attrList[selAttr][0]);
    setItemText(self, ATTR_TYPE, attrList[selAttr][1]);
    setItemText(self, ATTR_LENGTH, attrList[selAttr][2])
  else
    initEditBox(self)
  endif;

}       !!

/*set the current selection to idx */
Def setCurSel(self, idx)
{
  ^Call SendDlgItemMessage(hWnd,ATTR_LIST,LB_SETCURSEL,idx,0)
}      !!

Def  command(self, wP, lP)
{
   select
     case wP == IDCANCEL
       is setAttrList(parent, attrList);
          end(self, 0)
     endCase

     case wP == ATTR_DELETE
       is deleteAttr(self)
     endCase
```

```
        case wP == ATTR_TYPE
          is typeDialog := new(TypeDialog, typeList);
             runModal(typeDialog, ATTRLIST, self)
        endCase

        case wP == IDOK
          is addAttr(self)
        endCase

        case wP == ATTR_LIST
          is selItem(self)
        endCase
      endSelect

    }                        !!

    /* Initialize the listbox;  the method is the Actor
      equivalent of WM_INITDIALOG */
    Def  initDialog(self, wP, lP)
    {
      selAttr := nil;
      setItemText(self, OBJ_NAME, objName);
      do(attrList,
        {using(elem) insertString(self, subString(elem[0]
                            + "           ", 0,10) +  ": "
                            + elem[1] + "[" +
                            elem[2] + "]")});
      typeList := new(OrderedCollection, 15);
      add(typeList, tuple("Int", "S"));
      add(typeList, tuple("Money", "S"));
      add(typeList, tuple("String", "S"));
      typeList := addDispObj(parent, typeList);
    }                             !!
```

3. Bitmaps Class

```
/* this class class that retrieves bitmaps and displays them.
Bitmaps are stored in external files with .BMP extensions.
*/!!

inherit(DisplayOneWindow, #Bitmaps, #(bitStruct /* bitmap
dimensions*/
hBitmap /*handle to a bitmap*/
), 2, nil)!!
```

```
now(BitmapsClass)!!

now(Bitmaps)!!

/* this method reads in a bitmap struct. The long pointer to
global    memory that has been locked must be passed in, as
well as, the    file length. */

Def readInto2(self, file, lpGMem, numberofbits | start, aStr,
cnt)
{ start := 0;
  loop
  while (cnt :=  numberofbits - start) > 0
    cnt := min(cnt, 512);
    aStr := read(file, cnt);
    moveData(lP(aStr), lpGMem + asLong(start), cnt);
    freeHandle(aStr);
    start := start + cnt;
  endLoop;
}!!

/* this method calls the other methods to display desired
bitmap.    Bitname is the string name of the bitmap file to
be opened. */
Def showthepic(self,bitname)
{
  show2(self, 1, bitname);
} !!


/* draw the bitmap */
Def paint(self, hDC | hMemDC)
{ /* must get a compatible display context... */
  hMemDC := Call CreateCompatibleDC(hDC);
  /* ...and connect the bitmap to this DC.    */
  Call SelectObject(hMemDC, hBitmap);

  /* This allows us to copy it to the window. We have used the
Windows bitmap copy style NOTSRCCOPY which inverts the bitmap.
To  produce a bitmap which has the same colors use the Windows
style
SRCCOPY, 0xCC0020L, instead */

  Call BitBlt(hDC, 0, 0, wordAt(bitStruct, 2),
wordAt(bitStruct, 4),
    hMemDC, 0, 0, /* SRCCOPY */ 0xCC0020L);
  /* Since we created the second DC, we must delete it */
  Call DeleteDC(hMemDC);
}!!

/* this method opens the bitmap file and retrieves the bitmap
bits, the header information and creates a returns a handle
to a bitmap. */
Def loadbitfile(self, bitname | hfile,lpgMem,Filelen, bmHdr,
afile,
bmH,bmP,bmWB,numofbits,bmW,bmBPP,
     hgMem)
{    /* open the file named "bitname" and get a handle to it
*/
  if (afile := exists(File,bitname, 1)) then
     hfile := open(afile,2);
     Filelen := length(afile);
     moveTo(afile,0);
     if( length(afile) > 65535) then
        close(afile);
        setPicNil(parent); /* prevents dangling bitmap objects
on errors. */
       ^errorBox("SORRY", "file too large");
     endif;
  else
     setPicNil(parent);
     ^errorBox("SORRY", "Photo Not Avail");
  endif;

  /* create structure for the bitmap header information */
  bmHdr := new(Struct,16);

  readInto(bmHdr,afile);  /* reads in the header information
from                              the opened bitmap file. */

  bmW := wordAt(bmHdr,4);    /* bitmap width in bits */
  bmBPP := atMSB(bmHdr,10); /* bitmap bits per pixel */
  bmH := wordAt(bmHdr,6);    /* bitmap height in bits */
  bmP := atLSB(bmHdr,10);    /* bitmap number of Planes */
  bmWB := wordAt(bmHdr,8);  /* bitmap width of bytes */

 /* this calculates the actual number of bits stored in the
file       for the bitmap display.*/
  numofbits := bmH * bmP * bmWB;

  /* allocate memory space for the bitmap bits from the file
*/
```

```
  if (0 = (hgMem := Call GlobalAlloc(GMEM_MOVEABLE bitOr
GMEM_ZEROINIT,
        asLong(Filelen + 1)))) then
     close(afile);
     Call GlobalFree(hgMem);
     setPicNil(parent);
    ^errorBox("Error", "Cannot allocate memory for the bitmap
file");
   endif;

  lpgMem := Call GlobalLock(hgMem);
  /* retrieve the bits from the bitmap file */
  readInto2(self,afile,lpgMem,numofbits);

  /* create the bitmap */
  if (0 = ( hBitmap := Call CreateBitmap(bmW, bmH, bmP, bmBPP,
(lpgMem))))
   then
   Call GlobalFree(hgMem);
       setPicNil(parent);
      ^errorBox("Error", "Bitmap Empty");
 endif;

 Call GlobalFree(hgMem);
}!!

/* If the bitmap still exists when the window is destroyed,
   get rid of the bitmap and set picture object to nil */
Def WM_DESTROY(self, wp, lp)
{ if hBitmap
  then Call DeleteObject(hBitmap);
    hBitmap := nil;
  endif;
  setPicNil(parent);
  Call ReleaseCapture();
  ^WM_DESTROY(self:Window, wp, lp)
}!!


/* create the style for the bitmap windows. */
Def create(self, par, wName, rect, style| WS_BITMAP)
{
 WS_BITMAP :=  WS_POPUP bitOr WS_CAPTION bitOr
                    WS_SYSMENU bitOr WS_SIZEBOX;

 create(self:Window, par, "BITMAP", rect, WS_BITMAP);
}!!
```

```
 /* this method displays the bitmap */
Def show2(self, val, bitName | temp, height, width, x, y)
{
  parent := ThePort;
  loadbitfile(self,bitName);
  temp := new(Struct,14);
  /* exits if loadbitfile could not create a bitmap and return
a      handle. */
  if (nil = hBitmap) or (0 = hBitmap)
    ^0
  endif;

  /* get the information struct of the bitmap */
  Call GetObject(errorIfNil(hBitmap, #noPhoto), size(temp),
lP(temp));
  bitStruct := getData(temp);

  /* we are interested in the height and the width */
  height := wordAt(bitStruct, 4);
  width := wordAt(bitStruct, 2);
  /* compute the origin of the window, which will center
     the window */
  x :=(screenSize().x - width) / 2;
  y :=(screenSize().y - height) / 2;
   /* resize the window so that the bitmap fills the window
*/
  setCRect(self, rect(x, y, x+(width+5), y+(height+50)));
  moveWindow(self);
  show(self:WindowsObject, val);

}!!
```

4. DeleteWindow Class (Modified)

```
/* window to delete the instances of selObject */!!

inherit(MyWindow, #DeleteWindow, #(buttons  /*control
buttons*/
selObject /*glad object*/), 2, nil)!!

now(DeleteWindowClass)!!

now(DeleteWindow)!!

Def displayText(self | aStr)
{
  aStr := "Which data do you wish to delete?";
  Call TextOut(hDC,20,10,lP(aStr),size(aStr));
  freeHandle(aStr);
  aStr := asString(name(selObject))+" data being displayed";
  Call TextOut(hDC,100,42,lP(aStr),size(aStr));
  freeHandle(aStr);
  aStr := asString(name(selObject))+" data";
  Call TextOut(hDC,100,72,lP(aStr),size(aStr));
  freeHandle(aStr);
  aStr := asString(name(selObject))+" data";
  Call TextOut(hDC,100,102,lP(aStr),size(aStr));
  freeHandle(aStr)
}!!

Def help(self)
{
}!!

Def cancel(self)
{
}!!

Def delSelect(self)
{
}!!

Def delAll(self)
{
}!!

Def delCurrent(self)
{
}!!

Def start(self,obj)
{
  selObject := obj;
  createButtons(self);
  show(self,1)
}!!

Def initMenuID(self)
{
  /*this is actually a button IDs since this window has no
menu*/
  menuID := %Dictionary(
                              100->#delCurrent
```

```
                    101->#delAll
                    102->#delSelect
                    103->#help
                    104->#cancel
                    105->#close
                  )
}!!

Def paint(self,hdc)
{
  hDC := hdc;
  displayButtons(self);
  displayText(self)
}!!

Def createButtons(self)
{
  buttons := new(Array,6);
  buttons[0] := newPush(Button,100,self,"Current");
  buttons[1] := newPush(Button,101,self,"All");
  buttons[2] := newPush(Button,102,self,"Selected");
  buttons[3] := newPush(Button,103,self,"Help");
  buttons[4] := newPush(Button,104,self,"Cancel");
  buttons[5] := newPush(Button,105,self,"Quit");
}!!

Def displayButtons(self | left, top)
{
  left := 15;
  top  := 40;
  do(over(0,6), {using(idx)

setCRect(buttons[idx],rect(left,top,left+70,top+20));
                moveWindow(buttons[idx]);
                show(buttons[idx],1);
                top := top + 30 })
}!!

Def command(self,wp,lp)
{
  /*handles only the buttons; ignore edit controls*/
  if high(lp) <> 1 and menuID[wp]
    perform(self,menuID[wp])
  endif
}!!

Def create(self,par,wName,rect,style)
```

```
{
  ^create(self:Window,par,wName,&(200,150,550,400),
          WS_POPUP bitOr WS_CAPTION bitOr WS_SYSMENU)
}!!
```

5. DisplayOneWindow Class (Modified)

```
/* the class for displaying one member of selected object*/!!

inherit(OneMemWindow, #DisplayOneWindow, #(dispMemIdx /*index
of member              currently displayed */
members    /*instances of selObject*/
picture /* a bitmap object instance */
), 2, nil)!!

now(DisplayOneWindowClass)!!

now(DisplayOneWindow)!!

Def prev(self)
{
  if dispMemIdx == 0
    errorBox("WAIT","No more previous data")
  else
    dispMemIdx := dispMemIdx - 1;
    displayValues(self);

hiLiteNewMem(hasSibling(selObject,ListMemWindow),dispMemIdx)
  endif;
 }!!


Def allAtOnce(self | win)
{
  if dispMemIdx
    if win:=hasSibling(selObject,ListMemWindow)
      Call BringWindowToTop(handle(win));
      hiLiteNewMem(win,dispMemIdx)
    else
      win := new(ListMemWindow,parent,"GladLMMenu",
                 "DISPLAY: "+name(selObject),nil);
      addWindow(selObject,win);
      start(win,selObject);
      hiLiteNewMem(win,dispMemIdx)
    endif
  else
    errorBox("ERROR!","No member is highlighted")
  endif
}!!



/* this method sets the Bitmaps Pic object to nil to prevent
   multiple display of bitmaps. */
Def setPicNil(self)
{
 picture := nil;
 ^0;
 }!!

/* add buttons for bitmaps if attributes exists */
Def addBmbuttons(self )
{
  do(attributes(selObject),
        {using(attr)
           if (attr[CLASS] = "Bitmap")
             bmCnt := bmCnt +1;
           endif});
  createBmbuttons(self)
 } !!

/*create the pushbuttons to display bitmaps. Button ids are
arbitrarily started at 200 and added to the menu. A button
is created for each bitmap attribute found in the record. */

Def createBmbuttons(self |
attrname,buttonid,idx,menuidx,showmemberPict)
{
  if bmCnt
    menuidx := 12;
    buttonid := 200;
    idx := 0;
    do(attributes(selObject),
       {using(attr)
          if (attr[CLASS] = "Bitmap")
            attrname := attr[NAME];
            bmButtons[idx]:=
newPush(Button,buttonid,self,attrname);
            add(menuID,buttonid,#showmemberPict);
            buttonid := buttonid + 1;
            idx := idx +1;
```

```
                endif});
    endif;
  }!!
```

```
/* show the bitmap for the button pushed, only one bitmap
picture may be open at any one time. This exception may be
excluded for systems with sufficient memory to handle multiple
bitmap displays.
   Modification:  remove the if-endif block. This will enable
multiple bitmaps.  If the system locks up when multiple
bitmaps are displayed, reinstate this block or set up a
counter to handle the number of pictures your system will
handle.*/

Def showmemberPict(self |astr,fieldcnt,bmStrPos)
{ /* the previous instance of a bitmap is not nil*/
  if picture <> nil
    ^errorBox("Warning", "Close Prev Picture")
  endif;
  bmStrPos := bmMenuID - 200;
  fieldcnt := ((size(attributes(selObject)) -bmCnt) +
bmStrPos);   astr := members[dispMemIdx][fieldcnt];
  picture := new(Bitmaps,ThePort,nil," ",nil);
  showthepic(picture,astr);
 }!!

/*displays the idx'th member*/
Def displayNewMem(self, idx)
{
  dispMemIdx := idx;
  displayValues(self)
}!!
/*create dispEdit controls for attribute values*/
Def createECs(self | fieldCnt)
{
  fieldCnt := (size(attributes(selObject))-bmCnt);
  editControl := new(Array,fieldCnt);
  do(over(0,fieldCnt),
     {using(idx)
       editControl[idx] := new(DispEdit,idx,self,
          WS_BORDER bitOr WS_CHILD bitOr ES_LEFT
          bitOr ES_MULTILINE;} )
}!!

Def WM_KILLFOCUS(self,wp,lp | dummyEC)
{
  do(editControl,
     {using(ec) if wp == handle(ec)
                    errorBox("CAN'T EDIT in DISPLAY WINDOW",
                          "Choose Modify from the Menu");
```

```
                ^0
              endif})
}!!

/*add a new instance from AddOneWindow*/
Def append(self,newInst)
{
  add(members,newInst);
  /*display it*/
  last(self)
}!!


Def close(self)
{ if selObject <> nil /* handles closing of child windows */
    removeWindow(selObject,self);
  endif;
 /* if selObject <> nil and selObject<> selObj(parent)
       and not(referenced(selObject))
    avail(colorTable(parent),color(selObject));
    setColor(selObject,WHITE_COLOR);
    reDraw(parent,selObject)
  endif;*/
  close(self:Window)
}!!


Def modifyMember(self)
{
}!!

Def deleteMember(self | delWin)
{
  delWin := new(DeleteWindow,self,nil,"DELETE:
"+asString(name(selObject)),nil);
  start(delWin,selObject)
}!!

/*brings up the AddOneWindow for adding instances*/
Def addMember(self | addOneWin)
{
  addOneWin := new(AddOneWindow,self,nil,
                  "ADD : "+name(selObject), nil);
  start(addOneWin,selObject)
}!!

Def loadMembers(self | fields, line, aFile, fieldCnt)
```

```
{
  aFile := new(TextFile);
  setName(aFile, memberFile(selObject));
  open(aFile,2); /*read-write*/
  fieldCnt := size(attributes(selObject));
  loop while (line:=readLine(aFile))
    if (line <> "") then
      fields := new(Array, fieldCnt);
      do(over(0,fieldCnt),
          {using(idx) fields[idx] :=
subString(line,0,indexOf(line,'&',0));
                        line := delete(line,0,size(fields[idx])+1)
} );
      add(members,fields);
    endif;
  endLoop
}!!

Def start(self, obj, idx)
{ selObject := obj;
  dispMemIdx := idx;
  setScrollRanges(self);
  bmButtons := new(Array,10);
  members := new(OrderedCollection,50);
  initTextMetrics(self);
  bmCnt := 0;
  getbmCnt(self);
  createBmbuttons(self);
  loadTemplate(self);
  createECs(self);
  loadMembers(self);
  show(self,1)
}!!

Def next(self)
{
  if dispMemIdx == size(members) -1
    errorBox("WAIT","No next previous data")
  else
    dispMemIdx := dispMemIdx + 1;
    displayValues(self);

hiLiteNewMem(hasSibling(selObject,ListMemWindow),dispMemIdx)

  endif
}!!
```

```
/*display field values of one member*/
Def displayValues(self | fieldCnt)
{
  fieldCnt := (size(attributes(selObject))-bmCnt);
  do(over(0,fieldCnt),
    {using(idx)
setText(editControl[idx],members[dispMemIdx][idx]) } )
}!!

Def help(self|aStr)
{aStr :=asciiz("Display One Window");
  pcall(Lib.procs[#GUIDANCESETCONTEXT],HGuide,
  lP(aStr),1);
  freeHandle(aStr);

} !!

Def goToIth(self | iP, i)
{
  iP := new(InputDialog, "GO TO",
            "Type in the position# of desired data","");
  if runModal(iP,INPUT_BOX,self) == IDOK
    i := asInt(getText(iP),10);
    if i < 1 or i > size(members)
      errorBox("ERROR","Out of Range"+CR_LF+"Must be in
1.."+asString(size(members)) )
    else
      dispMemIdx := i -1;
      displayValues(self);

hiLiteNewMem(hasSibling(selObject,ListMemWindow),dispMemIdx)

    endif
  endif
}!!

Def last(self)
{
  dispMemIdx := size(members) - 1;
  displayValues(self);
  hiLiteNewMem(hasSibling(selObject,ListMemWindow),dispMemIdx)

}!!

Def first(self)
{
  dispMemIdx := 0;
  displayValues(self);
  hiLiteNewMem(hasSibling(selObject,ListMemWindow),dispMemIdx)

}!!

Def initMenuID(self)
{
  menuID := %Dictionary( 1->#addMember
                         2->#deleteMember
                         3->#modifyMember
                         4->#prev
                         5->#next
                         6->#first
                         7->#last
                         8->#goToIth
                         9->#allAtOnce
                         950->#help
                         11->#close)
}!!
```

6. GladObj Class (Modified)

```
/* for storing Glad objects such as emp, dept, etc. */!!

inherit(Object, #GladObj, #(name
rect   /*drawing rect for the ojbect */
color /*to fill the box when
       selected*/
nesting /*true if it is a
         generalized object*/
attributes /*collection of name,
        class, and type(U or S or C)*/
refCnt /*reference count*/
thickBorder/*true if most recently
            selected object*/
memberFile/*contains tuple*/
assocWindows /*collection of its opened windows*/
templateFile /*file containing display format*/
), 2, nil)!!

now(GladObjClass)!!

now(GladObj)!!

/*check if windowType window is opened for this glad object*/
```

```
Def hasSibling(self,windowType)
{
  do(assocWindows,{using(win) if class(win) = windowType ^win
endif});
  /*not found*/
  ^nil
}!!

Def templateFile(self)
{
  if templateFile = ""
    ^nil
  else
    ^templateFile
  endif
}!!

/*is self nested?*/
Def nesting(self)
{
  ^nesting
}!!

/*reset the attributes of self to attrList from DDWindow*/
Def setAttr(self,attrList)
{
  attributes := attrList
}!!

Def reDefine(self,newName,newNesting)
{
  name := newName;
  nesting := newNesting
}
  !!

/*check whether the self's name is in the OrderedCollection
of GladObjects*/
Def nameAlreadyUsed(self,dbSchema)
{
  do(dbSchema,{using(obj) if name(obj) = name /*of self*/
^true endif});

  ^nil
}
  !!
```

```
/*set the values for newly created object. this is called from
DDWindow*/
Def initialize(self,newName,nest,newRect)
{
  name := newName;
  if nest=NESTED then nesting := new(OrderedCollection,10)
endif;
  rect := newRect;
  refCnt := 0;
  color := WHITE_COLOR
}!!

/*save the self's attributes to the file*/
Def saveAttr(self, aSchemaFile)
{
  do(attributes,{using(attr) write(aSchemaFile,attr[NAME]+"&"+


attr[CLASS]+"&"+

attr[LENGTH]+"&"+

attr[USER_DEF]+"&"+CR_LF)});
  write(aSchemaFile,"&&"+CR_LF)
}
  !!

/*save the self's nested objects to the file*/
Def saveNesting(self, aSchemaFile)
{
  do(nesting, {using(aGladObj) save(aGladObj,aSchemaFile)});
  write(aSchemaFile,"@@"+CR_LF)
}
  !!

/*save the self to the file*/
Def save(self, aSchemaFile)
{
  write(aSchemaFile,name+CR_LF);  /*obj name*/
  write(aSchemaFile,asString(left(rect))+"@"+
                    asString( top(rect))+CR_LF); /*rect
location*/

  saveAttr(self,aSchemaFile);

  memberFile := memberFile cor " "; /*if undefined, save
blank*/
```

```
  write(aSchemaFile,memberFile+CR_LF);

  templateFile := templateFile cor " ";
  write(aSchemaFile,templateFile+CR_LF);

  if nesting /*save the nested objects*/
    write(aSchemaFile,"G"+CR_LF);
    saveNesting(self,aSchemaFile)
  else
    write(aSchemaFile,"N"+CR_LF)
  endif
}
  !!

Def removeWindow(self,win)
{
  remove(assocWindows,find(assocWindows,win))
}!!

Def name(self)
{
  ^name
}!!

Def rect(self)
{
  ^rect
}!!

Def memberFile(self)
{
  ^memberFile
}
!!

Def addWindow(self,win)
{
  add(assocWindows,win)
}!!

Def attributes(self)
{
  ^attributes
}  !!

Def closeOpenWindows(self)
{
  do(assocWindows, {using(win) close(win)})
} !!

/*obj is moved, adjust its rect*/
Def setNewRect(self, point, prevPt)
{
  offset(rect,x(point)-x(prevPt),
              y(point)-y(prevPt))
}   !!

Def thickBorder(self)
{
  thickBorder := true
} !!

Def regBorder(self)
{
  thickBorder := nil
} !!

Def setColor(self, aColor)
{
  color := aColor
} !!

Def color(self)
{
  ^color
} !!

/*see if self is referenced by others
  describe windows */
Def referenced(self)
{
  ^(refCnt > 0)
} !!

/*see if self has any opened window*/
Def anyOpenWindow(self)
{
  ^size(assocWindows) <> 0
}  !!

Def containedIn(self,point)
{
  ^(left(rect) <= x(point) and
      x(point) <= right(rect) and
```

```
    top(rect) <= y(point) and
     y(point) <= bottom(rect))
}
    !!

/*returns an inner box for a generalized object*/
Def nestedRect(self | tmpRect)
{
  tmpRect:=new(Rect);
  init(tmpRect,left(rect),top(rect),right(rect),bottom(rect));

  ^inflate(tmpRect,-5,-5)
}    !!

/*gets the nested objects and assign them to
  self's nesting */
Def getNesting(self, aSchemaFile, rectSize | anObj)
{
  nesting := new(OrderedCollection,5);
  anObj := new(GladObj);
  loop
  while get(anObj,aSchemaFile,rectSize)
    add(nesting,anObj);
    anObj := new(GladObj)
  endLoop
}   !!

/*reads in the next object from the schema file.
  rectSize is Point with width@height */
Def get(self, aSchemaFile, rectSize  | pt)
{
  if ( (name := readLine(aSchemaFile)) <> "@@")
    /*there's more*/
    pt := asPoint(readLine(aSchemaFile));
    rect:= new(Rect);

init(rect,x(pt),y(pt),x(pt)+x(rectSize),y(pt)+y(rectSize));
    attributes:= getAttr(self,aSchemaFile);
    memberFile:= readLine(aSchemaFile);
    templateFile:=readLine(aSchemaFile);
    if (at(readLine(aSchemaFile),0)=='G')
      getNesting(self,aSchemaFile,rectSize)
    endif;
    refCnt := 0;
    assocWindows := new(OrderedCollection,4);
    color := WHITE_COLOR; /*unselected color*/
    /*returns self*/
  else
    ^nil
  endif
}   !!

/*get the object attributes */
Def getAttr(self, aSchemaFile | aColl, anAttr, aStr)
{
  aColl := new(OrderedCollection,10);
  loop
  while ( (aStr := readLine(aSchemaFile)) <> "&&" )
    anAttr := new(Array,4);
    do (over(NAME,USER_DEF+1),
        {using(idx)
          anAttr[idx] :=
              subString(aStr,0,indexOf(aStr,'&',0));
          aStr := delete(aStr,0,size(anAttr[idx])+1)});
     add(aColl,anAttr)
  endLoop;
  ^aColl
}  !!
```

7. GladWindow Class (Modified)

```
/* top-level display window for GLAD */!!

inherit(MyWindow, #GladWindow, #(dbList/* DBDialog*/
openedDBs /*orderedcollection of            dbnames*/), 2,
nil)!!

now(GladWindowClass)!!

now(GladWindow)!!

Def close(self)
{ if dbList /*something is loaded*/
    updateDbsFile(dbList)
  endif;
  close(self:Window)
}!!

/* Initialise a call to Guidance */
Def  initGuidance(self|aStr,aString)
{Lib := new(Library);
  Lib.name := "Gydance.exe";
  add(Lib, #GUIDANCEINITIALISE, 0, #(0 0 1 1 0 0));
  add(Lib, #GUIDANCESETCONTEXT, 0, #(0 1 0));
  add(Lib, #GUIDANCETERMINATE, 0, #(0));
  load(Lib);
  aString := "GLAD";
  aStr := "index.gui";
  HGuide :=pcall(Lib.procs[#GUIDANCEINITIALISE],
  HInstance, handle(self), lP(aString),
  lP(aStr), 1,1);
}!!

/*initialize the variables; remove the scroll bar*/
Def init(self)
{
  dbList := new(DBDialog);
  openedDBs := new(OrderedCollection,2);
  setScrollRange(self,SB_VERT,0,0);
  setScrollRange(self,SB_HORZ,0,0);
  initMenuID(self)
}  !!

/*adds a new database name to a DBDialog object*/
Def addDb(self, dbName)
{
  addDb(dbList,dbName)
}!!

Def modifyDb(self | dbName, dDWin)
{
  setState(dbList, OPEN_DB);
  if runModal(dbList,OPNDBLIST,self) == OPEN_DB

    dbName := getSelDb(dbList);
    if size(extract(openedDBs,{using(db) db = dbName})) > 0
      errorBox("ERROR","Database '"+dbName+"' is already
opened.        "+
                         "Cannot open more than one data
definition        or "+
                         "data manipulation window for a single
    database.")
    else /*okay, open DD window*/
      add(openedDBs,dbName);
      dDWin := new(DDWindow,self,"GladDdlMenu",
                        "Modify Database: "+dbName,nil);
      startWithExistingDb(dDWin,dbName)
    endif
  endif
}            !!
```

```
Def initMenuID(self)
{
  menuID := %Dictionary ( 1->#makeNewDb
                          2->#modifyDb
                          3->#openDb
                          4->#removeDb
                          950->#topHelp
                          6->#close )
}  !!

/*create it as overlapped window; need for stand-alone appl*/
Def create(self,par,wName,rect,style)
{
  ^create(self:MyWindow,nil,wName,rect,WS_OVERLAPPEDWINDOW)
} !!

Def removeDb(self)
{
  setState(dbList,REMOVE_DB);
  runModal(dbList,RMVDBLIST,self)
}   !!

Def openDb(self | dMWin , dbName)
 {
  setState(dbList,OPEN_DB);
  if runModal(dbList,OPNDBLIST,self) == OPEN_DB

    dbName := getSelDb(dbList);
    if size(extract(openedDBs,{using(db) db = dbName})) > 0
      errorBox("ERROR","Database '"+dbName+"' is already
opened.        "+
                         "Cannot open more than one data
definition        or "+
                         "data manipulation window for a single
     database.")
    else /*okay, open DM window*/
      add(openedDBs,dbName);
      dMWin := new(DMWindow, self, "GladDmlMenu",
                   "Data Manipulation: "+dbName,nil);

     start(dMWin,dbName);
    endif
  endif
}                  !!

Def topHelp(self|aStr)
```

```
{aStr :=asciiz("GLAD WINDOW");
  pcall(Lib.procs[#GUIDANCESETCONTEXT],HGuide,
  lP(aStr),1);
  freeHandle(aStr);
} !!

Def makeNewDb(self | inpDbNameDlg, dbName, dDWin)
{
  inpDbNameDlg := new(InputDialog, "Create Database",
                "Enter the name for a new database ", "");

  if runModal(inpDbNameDlg, INPUT_BOX, self) == IDOK

    dbName := getText(inpDbNameDlg);
    if nameExist(dbList,dbName)  /* Database name already
exists */
      errorBox("ERROR!", "Database '" + dbName + "' already
exists!")
    else      /*  okay, can use this dbName */
              /*  start the DDwindow */
      dDWin := new(DDWindow,self,"GladDdlMenu",
                    "Define Database: "+dbName+"
(*NEW*)",nil);
      startWithNoDb(dDWin,dbName)
    endif
  endif;

}               !!
```

8. MyWindow Class (Modified)

```
/* Special window type for Glad application.  This class
   renames some pass-along methods, interprets SCROLL
messages, defines new command, etc. */!!

inherit(Window, #MyWindow, #(menuID  /*menu id numbers,
1,2,...*/
rbuttonDn /*state of right button*/
hDC      /*display context*/
hScrollID /*horz scroll ids, 0,1,..*/
vScrollID /*vert scroll ids, 0,1,..*/
), 2, nil)!!

now(MyWindowClass)!!

now(MyWindow)!!
```

```
Def getScrollPos(self,bar)
{
  ^Call GetScrollPos(hWnd,bar)
} !!

Def command(self,wp,lp)
{/*only interprets the menu choice now*/
  if menuID[wp] and high(lp) = 1
    perform(self,menuID[wp]);
  else
    if menuID[wp]
      perform(self,menuID[wp])
    else ^0
    endif;
  endif;
}!!

Def WM_VSCROLL(self,wp,lp)
{
  if vScrollID[wp] /*defined, so do it*/
    perform(self,lp,vScrollID[wp])
  else
    ^0
  endif
}  !!

Def WM_HSCROLL(self,wp,lp)
{
  if hScrollID[wp] /*if defined, then perform it*/
    perform(self,lp,hScrollID[wp])
  else
    ^0
  endif
}    !!

/*called from new method to initialize menu and scroll IDs*/
Def init(self)
{
  initMenuID(self);
  inithScrollID(self);
  initvScrollID(self);
}   !!

Def initvScrollID(self)
{
  vScrollID := new(Dictionary,5);
  add(vScrollID,SB_LINEUP,#upArrow);
  add(vScrollID,SB_LINEDOWN,#downArrow);
  add(vScrollID,SB_PAGEUP,#upPage);
  add(vScrollID,SB_PAGEDOWN,#downPage);
  add(vScrollID,SB_THUMBPOSITION,#vThumbPos)
} !!

Def inithScrollID(self)
{
  hScrollID := new(Dictionary,5);
  add(hScrollID,SB_LINEUP,#leftArrow);
  add(hScrollID,SB_LINEDOWN,#rightArrow);
  add(hScrollID,SB_PAGEUP,#leftPage);
  add(hScrollID,SB_PAGEDOWN,#rightPage);
  add(hScrollID,SB_THUMBPOSITION,#hThumbPos)
} !!

Def initMenuID(self)
{
  menuID := %Dictionary()
}  !!

Def setScrollRange(self,bar,min,max)
{
  Call SetScrollRange(hWnd,bar,min,max,1)
} !!

Def setScrollPos(self,bar,pos)
{
  Call SetScrollPos(hWnd,bar,pos,1)
} !!

/*MyWindow has vert and horz scroll bars*/
Def create(self,par,wName,rect,style)
{
  create(self:Window,par,wName,rect,
          style bitOr WS_HSCROLL bitOr WS_VSCROLL)
} !!

Def rButtonRelease(self,wp,point)
{
  ^0
} !!

Def lButtonRelease(self,wp,point)
{
  ^0
} !!
```

```
Def beginDrag(self,wp,point)
{
  lButtonDown(self,wp,point)
} !!

Def endDrag(self,wp,point)
{
  lButtonRelease(self,wp,point)
}  !!

Def lButtonDown(self,wp,point)
{
  ^0
} !!

Def rButtonDown(self,wp,point)
{
  ^0
} !!

/* MS Windows message for right button down*/
Def WM_RBUTTONDOWN(self,wp,lp)
{
  if rbuttonDn
    ^0
  endif;
  rbuttonDn := true;
  Call SetCapture(hWnd);
  rButtonDown(self,wp,asPoint(lp))
} !!

/* MS Windows message for right button release*/
Def WM_RBUTTONUP(self,wp,lp)
{
  if not(rbuttonDn)
    ^0
  endif;
  rbuttonDn := nil;
  Call ReleaseCapture();
  rButtonRelease(self,wp,asPoint(lp))
}  !!

/* mouse is moved while the left button is down*/
Def mouseMoveWithLBDn(self,wp,point)
{
  ^0
} !!

Def drag(self,wp,point)
{
  mouseMoveWithLBDn(self,wp,point)
} !!
```

9. NestDMWindow Class (modified)

```
/* A nested DMWindow for displaying nested objects */!!

inherit(DMWindow, #NestDMWindow, #(genObj /*generalized object
of this windows' nested objects*/), 2, nil)!!

now(NestDMWindowClass)!!

now(NestDMWindow)!!

/* draws an object on the window using
   the hdc display context ; modified from its parent
   so the rect is drawn with the genobj's color*/
Def display(self,obj,hdc | objName, objRect,
                              hBrush, hPen, hOldBrush, hOldPen)
{
  eraseRect(self,obj,hdc); /*first erase it*/
  /*select the color brush for filling
    used with Rectangle (via draw) */
  hBrush := Call CreateSolidBrush(color(obj));
  /*set bkcolor for shading with DrawText*/
  Call SetBkColor(hdc,color(obj));
  hOldBrush := Call SelectObject(hdc,hBrush);
  objRect := rect(obj);
/*  if hasThickBorder(obj) draw it with a thick border*/
    hPen := Call CreatePen(0,5,color(genObj));
 /* else */               /*draw it with a regular border*/
  /*  hPen := Call CreatePen(0,1,color(genObj))
  endif;*/
  hOldPen:= Call SelectObject(hdc,hPen);
  draw(objRect,hdc);
  Call SelectObject(hdc,hOldPen);/*restore the dc*/
  Call DeleteObject(hPen);

  if nesting(obj) /*draw the inner box if it is a nested
object*/
    hPen := Call CreatePen(0,1,color(genObj));
    hOldPen:= Call SelectObject(hdc,hPen);
    draw(nestedRect(obj),hdc);
    Call SelectObject(hdc,hOldPen);/*restore the dc*/
    Call DeleteObject(hPen)
  endif;

  objName := name(obj);
  Call DrawText(hdc,lP(objName),-1,objRect,
              DT_CENTER bitOr DT_VCENTER
              bitOr DT_SINGLELINE);
  Call SelectObject(hdc,hOldBrush);
  Call DeleteObject(hBrush);
  freeHandle(objName)
}   !!

Def close(self)
{
  removeWindow(genObj, self);
  do(dbSchema,{using(obj) if color(obj)
                          avail(colorTable,color(obj));
                          setColor(obj,WHITE_COLOR)
                        endif } );
  close(self:Window)
}
  !!

Def start(self,obj,colorTbl | nullStr)
{
  genObj := obj;
  dbSchema := nesting(obj);
  colorTable := colorTbl;
  nullStr := "";
  changeMenu(self,
9/*showConnection*/,lP(nullStr),0,MF_DELETE);
freeHandle(nullStr);
  drawMenu(self);
  show(self,1)
}!!
```

10. OneMemWindow Class (Modified)

```
/* a formal window for listing one member of an object.
   inherited by DisplayOneWindow and QueryWindow */!!

inherit(MyWindow, #OneMemWindow, #(selObject /*selected
object*/
editControl /*array of ECs*/
template  /*display layout*/
tmWidth   /*char width*/
tmHeight  /*char height*/
aLMWin     /*sibling listMemWin*/
bmCnt      /* bitmap count number*/
bmButtons /* bitmap array of buttons */
bmMenuID  /* bitmap menu identifier */
), 2, nil)!!

now(OneMemWindowClass)!!

now(OneMemWindow)!!

/* displays the bitmap buttons.  The user has an option of
displaying the buttons horizontally or vertically.
DisplayBmbuttons2 displays the buttons horizontally.
DisplayBmbuttons3 displays the buttons vertically. */
Def displayBmbuttons3(self |leftx,left,top,fieldcnt)
{
  fieldcnt := (size(attributes(selObject)) - bmCnt -1);
  left :=  tmWidth ;
  top := ((template[fieldcnt][EC_LOC].y + 3) * tmHeight) +
         template[fieldcnt][EC_DIM].y;
  do(over(0,(bmCnt)),
       {using(idx)

setCRect(bmButtons[idx],rect(left,top,left+70,top+20));
    moveWindow(bmButtons[idx]);
         show(bmButtons[idx],1);
         top := top + 30 });
 }!!

/* displays the bitmap buttons */
Def displayBmbuttons2(self |leftx,left,top,fieldcnt)
{
  fieldcnt := (size(attributes(selObject)) - bmCnt -1);
  leftx:= template[fieldcnt][EC_LOC].x +
template[fieldcnt][EC_DIM].x;
  left := (leftx * tmWidth) +40;
  top := template[fieldcnt][EC_LOC].y * tmHeight;
  do(over(0,(bmCnt)),
       {using(idx)

setCRect(bmButtons[idx],rect(left,top,left+70,top+20));
    moveWindow(bmButtons[idx]);
         show(bmButtons[idx],1);
         top := top + 30 });
```

```
}!!

/* returns the bitmap identifier, wcrd parameter. */
Def getbmMenuID(self,wp)
{
  ^bmMenuID := wp;
}!!


/* get the bitmap count for the selected object */
Def getbmCnt(self )
{ bmCnt := 0;
  do(attributes(selObject),
       {using(attr)
          if (attr[CLASS] = "Bitmap")
            bmCnt := bmCnt +1;
          endif});
}!!

/*redefines the command to handle non-menu message and
buttons*/ Def command(self,wp,lp)
{
  if wp > 199 and menuID[wp]
    getbmMenuID(self,wp);
    perform(self,menuID[wp])
  endif;
  if lp = 0  /*menu*/
    command(self:MyWindow,wp,lp)
  else
    /*do nothing for now*/
    ^0
  endif
}!!

/*prints out the attribute label and its edit box*/
Def printAttr(self,attr,idx |bmClass, aLabel, ECx, ECy)
{
  aLabel := attr[NAME];
  if ((bmClass := attr[CLASS]) <> "Bitmap")
    Call TextOut(hDC, template[idx][LABEL_LOC].x * tmWidth,
                      template[idx][LABEL_LOC].y * tmHeight,
                      lP(aLabel), size(aLabel));
    freeHandle(aLabel);
    ECx := template[idx][EC_LOC].x * tmWidth;
    ECy := template[idx][EC_LOC].y * tmHeight;
    setCRect(editControl[idx],
            rect(ECx, ECy,
                 ECx + (template[idx][EC_DIM].x + 3) *
tmWidth,
                 ECy + asInt((template[idx][EC_DIM].y+0.5) *
tmHeight)));
    moveWindow(editControl[idx]);
    show(editControl[idx],1);
  endif;
}!!

Def setScrollRanges(self)
{
  setScrollRange(self,SB_HORZ,0,0); /*hide them*/
  setScrollRange(self,SB_VERT,0,0)
}!!

/*load the template. use default setup if
  the template file is not defined */
Def loadTemplate(self | aFile, fieldCnt, tmp, dimension,
attrLength,
                       yloc, ecHeight, ecWidth, attribute)
{
  fieldCnt := (size(attributes(selObject))-bmCnt);
  template :=new(Array,fieldCnt);

  if templateFile(selObject) /*defined*/
    aFile := new(TextFile);
    setName(aFile,templateFile(selObject));
    open(aFile,0);
    do( over(0,fieldCnt),
        {using(idx)  tmp := new(Array,3);
                     tmp[LABEL_LOC] :=
asPoint(readLine(aFile));
                     tmp[EC_LOC]     :=
asPoint(readLine(aFile));
                     tmp[EC_DIM]     :=
asPoint(readLine(aFile));
                     template[idx]  := tmp } );
    dimension := asPoint(readLine(aFile))
  else
    /*template not defined; use a default layout*/
    yloc := 1; attribute := attributes(selObject);
    do(over(0,fieldCnt),
        {using(idx)  tmp := new(Array,3);
                     tmp[LABEL_LOC] := point(1,yloc);
                     tmp[EC_LOC]     := point(1,yloc+1);
                     attrLength :=
asInt(attribute[idx][LENGTH],10);
```

```
                    ecWidth := min(20,attrLength);
                    ecHeight := attrLength/20 + 1;
                    tmp[EC_DIM]     :=
point(ecWidth,ecHeight);
                    template[idx]  := tmp;
                    yloc := yloc + 3 + ecHeight } );
    dimension := point(30,yloc + 10/*for menu & title*/)
  endif;

setCRect(self,rect(100,100,100+dimension.x*tmWidth,100+dimen
sion.y*tmHeight));
  moveWindow(self)
}!!

Def paint(self,hdc)
{
  hDC := hdc;
  displayTemplate(self);
  displayValues(self);
  if bmCnt and bmButtons <> nil
   /* replace with ..Bmbuttons2 to display buttons
horizontally*/
    displayBmbuttons3(self)
  endif;
}!!

/*display attribute values. dummy method for this class*/
Def displayValues(self)
{
}!!

/*lay out the display format*/
Def displayTemplate(self | idx, color)
{
  idx := 0;
  /* set up the text color for textOut*/
  color := Call SetTextColor(hDC, color(selObject));

  do(attributes(selObject),
    {using(attr) printAttr(self,attr,idx);
                 idx := idx + 1 } );

  /* reset the text color*/
  Call SetTextColor(hDC, color)
}!!

/*create edit controls for attribute values*/
```

```
Def createECs(self | fieldCnt)
{
  fieldCnt := (size(attributes(selObject))-bmCnt);
  editControl := new(Array,fieldCnt);
  do(over(0,fieldCnt),
     {using(idx)
       editControl[idx] := new(Edit,idx,self,
          WS_BORDER bitOr WS_CHILD bitOr ES_LEFT
          bitOr ES_MULTILINE)} )
}!!

/* initialize the char width and height for this window*/
Def initTextMetrics(self | hdc, tm)
{
  tm := new(Struct,32);
  Call GetTextMetrics(hdc:=Call GetDC(hWnd), tm);
  tmWidth := asInt(wordAt(tm,10));
  tmHeight:= asInt(wordAt(tm,8)) + asInt(wordAt(tm,0));
  Call ReleaseDC(hWnd,hdc)
}!!
```

APPENDIX B. GLAD DATABASES FILE

```
/* GLAD databases */

AH1S Helicopter DB
Leisure Planning
NPS Lab Equipment
Pine Valley Furniture Co.
Test Connection DB
University Database
```

APPENDIX C. AH1S SCHEMA FILE

```
/* Schema file for the AH1S database */

ELEC_SYS
100@100
Description&String&35&S&
FSN&String&17&S&
Part_Number&String&25&S&
SMR&String&10&S&
FSCM&String&10&S&
U/M&String&5&S&
QTY&String&10&S&
PICTURE&Bitmap&25&S&
&&
elecsys.dat
elecsys.tpl
N
COBRA PHOTOS
250@100
_&String&40&S&
PICTURE1&Bitmap&25&S&
PICTURE2&Bitmap&25&S&
PICTURE3&Bitmap&25&S&
&&
cobra.dat
cobra.tpl
N
FUEL_SYS
400@200
Description&String&35&S&
FSN&String&17&S&
Part_Number&String&25&S&
SMR&String&16&S&
FSCM&String&16&S&
U/M&String&7&S&
QTY&String&6&S&
PICTURE&Bitmap&25&S&
&&
fuelsys.dat
fuelsys.tpl
N
FLT_CTL_SYS
400@100
Description&String&35&S&
FSN&String&17&S&
```

```
Part_Number&String&25&S&
SMR&String&16&S&
FSCM&String&20&S&
U/M&String&7&S&
QTY&String&6&S&
PICTURE&Bitmap&30&S&
&&
fctlsys.dat
fctlsys.tpl
G
CTL_STICK_ASSY
400@100
Description&String&35&S&
FSN&String&17&S&
Part_Number&String&25&S&
SMR&String&16&S&
FSCM&String&20&S&
U/M&String&7&S&
QTY&String&6&S&
PICTURE&Bitmap&30&S&
&&
fctlsys2.dat
fctlsys2.tpl
N
@@
ENV_SYS
100@200
Description&String&35&S&
FSN&String&17&S&
Part_Number&String&25&S&
SMR&String&20&S&
FSCM&String&10&S&
U/M&String&7&S&
QTY&String&6&S&
PICTURE&Bitmap&25&S&
&&
envsys.dat
envsys.tpl
N
AIR_FM_TOOLS
250@200
Description&String&35&S&
BOI&String&25&S&
FSN&String&17&S&
Part_Number&String&25&S&
SMR&String&20&S&
FSCM&String&16&S&
```

```
U/M&String&7&S&
QTY&String&6&S&
PICTURE&Bitmap&25&S&
&&
aftools.dat
aftools.tpl
G
SPECIAL_TOOLS
250@200
Description&String&35&S&
BOI&String&25&S&
FSN&String&17&S&
Part_Number&String&25&S&
SMR&String&10&S&
FSCM&String&10&S&
U/M&String&7&S&
QTY&String&6&S&
PICTURE&Bitmap&25&S&
&&
sptools.dat
sptools.tpl
N
@@
@@
```

APPENDIX D. AH1S DATA FILE

/* Data for the AH1S database */

1. COBRA.DAT File

```
AH1S COBRA ATTACK HELICOPTER&bitmaps\ahmain2.BMP&
bitmaps\AHMAIN.BMP&bitmaps\AHMAIN.BMP&
```

2.ELECSYS.DAT (Electrical System)

```
PANEL,POWER DIST BRKR&6110-00-137-2347&209-075-220-15&XBOZZ
&97499&EA&1&bitmaps\AH195K.BMP&
1 NUT, SELF LOCKING,EX&5310-00-807-1474&MS21042L3&PAOZZ&96906
&EA&10&bitmaps\AH195K.BMP&
1A WASHER,FLAT&5310-00-167-0834&AN960-10L&PAOZZ&96906&EA&10
&bitmaps\AH195K.BMP&
1B SCREW,MACHINE&5305-00-989-7434&MS35207-263&PAOZZ &96906&
EA&10 &bitmaps\AH195K.BMP&
2 CABLE ASSEMBLY&NONE&209-075-415-19&AOOOO&97499&EA&1
&bitmaps\AH195K.BMP&
2A TERMINAL, LUG&5940-00-143-5284&MS25036-115&PAOZZ&96906&EA&8
&bitmaps\AH195K.BMP&
3 CABLE ASSEMBLY&NONE&209-075-415-9&AOOOO&97499&EA&1
&bitmaps\AH195K.BMP&
3A TERMINAL,
LUG&5940-00-143-5284&MS25036-115&PAOZZ&96906&EA&10
&bitmaps\AH195K.BMP&
3B TERMINAL, LUG&5940-00-114-1305&MS25036-116&PAOZZ&96906&EA&2
&bitmaps\AH195K.BMP&
4 BUS, CONDUCTOR&6150-00-828-3094&204-075-230-15&PAOZZ&97499&
EA&2&bitmaps\AH195K.BMP&
5 BUS, CONDUCTOR&NONE&204-075-230-9&PAOZZ&97499&EA&3&
bitmaps\AH195K.BMP&
5A BUS, BAR&NONE&204-075-230-9&PAOZZ&97499&EA&1&
bitmaps\AH195K.BMP&
6 BUS, CONDUCTOR&NONE&204-075-230-3&PAOZZ&97499&EA&1&
bitmaps\AH195K.BMP&
6A BUS, BAR USE AFT&NONE&204-075-230-3&PAOZZ&97499&EA&3&
bitmaps\AH195K.BMP&
7 BUS, CONDUCTOR&6150-00-984-5362&204-075-230-13&PAOZZ&97499&
EA&1&bitmaps\AH195K.BMP&
8 BUS, BAR&NONE&204-075-230-7&MOOZZ&97499&EA&1&
bitmaps\AH195K.BMP&
8A BUS, BAR, USE AFT&NONE&204-075-230-7&MOOZZ&97499&EA&1&
bitmaps\AH195K.BMP&
```

9 BUS, CONDUCTOR&6150-00-225-0267&204-075-230-13&PAOZZ&97499&
EA&1&bitmaps\AH195K.BMP&
10 BUS, CONDUCTOR, OBS AFT&NONE&204-075-230-5&PAOZZ&97499&
EA&1&bitmaps\AH195K.BMP&
11 CIRCUIT BREAKER&5925-00-985-8319&MS22073-5&PAOZZ&96906&
EA&29&bitmaps\AH195J.BMP&
11A CIRCUIT BREAKER,USE AFT&5925-00-850-7244&MS24509-5-5&
PAOZZ&96906&EA&1&bitmaps\AH195J.BMP&
12 CIRCUIT BREAKER&5925-00-937-5669&MS22073-10&PAOZZ&96906&
EA&8&bitmaps\AH195J.BMP&
13 CIRCUIT BREAKER&5925-00-059-1140&MS22073-7
1-2&PBOZZ&96906&EA&1& bitmaps\AH195J.BMP&
13A CIRCUIT BREAKER, USE AFT&5925-00-079-2930&MS24509-7&PAOZZ&
96906&EA&1&bitmaps\AH195J.BMP&
14 CIRCUIT BREAKER&5925-00-686-3301&MS25244-15&PAOZZ&
96906&EA&6&bitmaps\AH195J.BMP&
14A CIRCUIT BREAKER, USE AFT&5925-00-842-7298&MS24509-15&
PAOZZ&96906&EA&3&bitmaps\AH195J.BMP&
15 CIRCUIT BREAKER&5925-00-912-8343&MS22073-1&PAOZZ&96906&
EA&1&bitmaps\AH195J.BMP&
16 CIRCUIT BREAKER&5925-00-805-2833&MS25244-25&PAOZZ&96906&
EA&1&BITMAPS\AH195J.BMP&
17 PLATE, IDENTIFICATION&9905-00-442-2576&209-075-222-1&PBOZZ&
97499&EA&1&BITMAPS\AH195J.BMP&
18 PLATE, IDENTIFICATION&9905-00-178-4804&209-075-222-3&PBOZZ&
97499& EA&1&BITMAPS\AH195J.BMP&
19 PLATE, IDENTIFICATION&9905-00-178-5849&209-075-222-47&
XBOZZ&97499&EA&1&BITMAPS\AH195J.BMP&
20 PLATE, IDENTIFICATION&9905-00-442-2578&209-075-222-9&
PBOZZ&97499&EA&1&BITMAPS\AH195J.BMP&
21 PLATE, MARKING,BLANK&9905-00-178-4802&209-075-222-75&
PBOZZ&97499&EA&1&BITMAPS\AH195J.BMP&
22 PLATE, IDENTIFICATION&9905-00-138-8689&209-075-222-53&
XBOZZ&97499&EA&1&BITMAPS\AH195J.BMP&
23 PLATE, SPECIAL&9905-00-151-7933&209-075-222-45&XBOZZ&
97499&EA&1&BITMAPS\AH195J.BMP&
24 PLATE, IDENTIFICATION&9905-00-234-7011&209-075-222-55&
XBOZZ&97499&EA&1&BITMAPS\AH195J.BMP&
25 PLATE, IDENTIFICATION&NONE&209-075-222-41&XBOZZ&97499&
EA&1&BITMAPS\AH195J.BMP&
26 PLATE, IDENTIFICATION&9905-00-178-4806&209-075-222-39&
PBOZZ&97499&EA&1&BITMAPS\AH195J.BMP&
27 PLATE, IDENTIFICATION&9905-00-400-7285&209-075-222-29&
PBOZZ&97499&EA&1&BITMAPS\AH195J.BMP&
28 PLATE, IDENTIFICATION&9905-00-220-2070&209-075-222-31&
PBOZZ&97499&EA&1&BITMAPS\AH195J.BMP&
29 PLATE, ASSEMBLY&NONE&209-075-221-1&MFFZZ&97499&EA&1&

BITMAPS\AH195J.BMP&

3.ENVSYS.DAT File (Environment Control System)

EI-AIR DISTR,HEATING VENT&NONE&209-070-402-1&XC&97499&EA&1&
BITMAPS\BMAHa7b&
1 NUT,SELF-LOCKING,EX&5310-00-807-1474&MS21042L3&PAOZZ
&96906&EA&29&BITMAPS\BMAH195&
2 WASHER,FLAT&5310-00-167-0753&AN96OPD10L&PAOZZ&88044&
EA&26&BITMAPS\BMAH273&

4.FCTLSYS.DAT File (Flight Control System)

EI-GRIP ASSEMBLY CONTROL&1689-00569-9573&22228&PA000&95712&EA&

1&BITMAPS\AH247F.BMP&
1 GUARD U/O PN 21885&1680-00-450-1851&21911&PB0ZZ&95712&EA&1&
BITMAPS\AH247G.BMP&
2 SPRING,HELICAL U/O PN 21885-4&5360-00-451-0060&21912&PB0ZZ&
95712&EA&1&BITMAPS\AH247G.BMP&
3 SWITCH SUBASSEMBLY U/O PN 21885-4&5930-00-612-6993&21230&
PA0ZZ&77820&EA&1&BITMAPS\AH247F.BMP&
4 SCREW&NONE&PM21402&XD0ZZ&95712&EA&1&BITMAPS\AH247E.BMP&

5 BRACKET&NONE&PM21254&XD0ZZ&95712&EA&1&BITMAPS\AH247E.BMP&

CONTROL STICK ASSEMBLY&NONE&209-001-301-3&A000F&97499&
EA&1&BITMAPS\AH246C.BMP&
1 PIN COTTER&5315-00-815-1405&MS24665-151&PAOZZ&96906&
EA&1&BITMAPS\AH246D.BMP&
2 NUT SELF-LOCKING,SL&5310-00-900-9421&MS178825-5&PA0ZZ&
95712&EA&1&BITMAPS\AH246D.BMP&
3 WASHER,FLAT&5310-00-187-2399&21230&PA0ZZ&77820&EA&2&
BITMAPS\AH246D.BMP&
4 BOLT,CLOSE TOLERANCE&5306-00-180-0473&PAOZZ&XD0ZZ&95712&
EA&1&BITMAPS\AH246D.BMP&
5 SUPPORT ASSEMBLY&1560-00-917-1799&209-001-316-1&XD0ZZ&95712&
EA&1&BITMAPS\AH246D.BMP&
6 BEARING,BALL,AIRFRAME&3110-00-158-6298&DW5&PAOZZ&21335
&EA&1&BITMAPS\AH246D.BMP&
7 PIN,COTTER&5315-00-815-1405&MS24665-151&PAOZZ&96906&
EA&1&BITMAPS\AH246D.BMP&
8 NUT SELF-LOCKING,SL&5310-00-961-8390&MS178825-4&PA0ZZ&
96906&EA&1&BITMAPS\AH246D.BMP&
9 NUT SELF-LOCKING,EX&5310-00-807-1474&MS21042L3&PA0ZZ&96906&
EA&1&BITMAPS\AH246D.BMP&
9A WASHER,FLAT&5310-00-167-0753&AN960PD&PA0ZZ&88044&EA&1&

BITMAPS\AH246D.BMP&
9B SCREW,MACHINE&5305-00-989-7435&MS35207-264&PAOZZ&96906&
EA&1&BITMAPS\AH246D.BMP&
9C BRACKET,AIRCRAFT&NONE&20-032-3&MFOZZ&97499&EA&1&
BITMAPS\AH246D.BMP&
9D CLAMP, LOOP&5340-00-989-9224&MS25281R6&PAOZZ&81996&EA&1&
BITMAPS\AH246D.BMP&
10 WASHER,FLAT&5310-00-187-2354&AN960PD&PA0ZZ&88044&EA&2&
BITMAPS\AH246D.BMP&
11 BOLT,CLOSE TOLERANCE&5306-00-180-1744&PAOZZ&XD0ZZ&AN174-25&

EA&1&BITMAPS\AH246D.BMP&
12 BELL CRANK ASSEMBLY&1680-00-918-6387&209-001-318-1&PAOFZ&
97499&EA&1&BITMAPS\AH246D.BMP&
13 BEARING,PLANE,SELF-LOCK&3120-00-989-4043&209-001-051-1&
PAFZZ&97499&EA&1&BITMAPS\AH246D.BMP&
15 CONNECTOR,PLUG,ELEC&5935-00-724-7591&MS3126F14-19P&
PAOZZ&96906&EA&1&BITMAPS\AH246D.BMP&
16 NUT SELF-LOCKING,EX&5310-00-807-1474&MS21042L3&PA0ZZ&96906&
EA&1&BITMAPS\AH246D.BMP&
17 WASHER,FLAT&5310-00-183-4406&AN960PD10&PA0ZZ&88044&
EA&1&BITMAPS\AH246D.BMP&
18 SCREW,MACHINE&5305-00-944-5929&MS27039-1-07&PAOZZ&96906&
EA&1&BITMAPS\AH246D.BMP&
19 CLAMP, LOOP&5340-00-990-9301&EAB700D7&PAOZZ&81996&EA&1&
BITMAPS\AH246D.BMP&
20 NUT SELF-LOCKING,EX&5310-00-807-1474&MS21042L3&PA0ZZ&96906&
EA&1&BITMAPS\AH246D.BMP&
21 WASHER,FLAT&5310-00-183-4406&AN960PD&PA0ZZ&88044&EA&2&
BITMAPS\AH246D.BMP&
22 SCREW,MACHINE&5305-00-948-4152&MS27039-1-13&PAOZZ&96906&
EA&1&BITMAPS\AH246D.BMP&
23 SUPPORT, CLAMP&NONE&209-001-338-1&MFOZZ&97499&EA&1&
BITMAPS\AH246D.BMP&
24 GRIP ASSY,CTL,GUNNER BRKDOWN&1680-00-111-3024&
209-001-059-1&PAOOO&97499&EA&1&BITMAPS\AH246C.BMP&
25 NUT SELF-LOCKING&5310-00-902-6676&MS21083N3&PA0ZZ&96906&
EA&1&BITMAPS\AH246C.BMP&
25A SCREW,MACHINE&5305-00-655-6556&MS35266-70&PAOZZ&96906&
EA&1&BITMAPS\AH246C.BMP&
26 SWITCH,PUSH&5930-00-823-2115&MS25089-4AR&PAOZZ&96906&
EA&1&BITMAPS\AH246C.BMP&
26A SWITCH,PUSH U/O PN 702&5930-00-687-1973&MS25089-5AR&PAOZZ&
96906&EA&2&BITMAPS\AH246C.BMP&
27 PIN,STRAIGHT,HEADLESS&5315-00-148-3580&10-279-1&
PBOZZ&81579&EA&1&BITMAPS\AH246C.BMP&
28 SWITCH,TRIGGER&5930-00-172-9072&401-1301&XBOZZ&81579&

EA&1&BITMAPS\AH246C.BMP&
29 PIN,STRAIGHT,HEADLESS&5315-00-169-5089&STD-870-5&PBOZZ&
81579&EA&1&BITMAPS\AH246C.BMP&
30 SWITCH,TOGGLE&5930-00-013-9136&MS27708-3&PAOZZ&96906&
EA&1&BITMAPS\AH246C.BMP&
31 CONTROL STICK, CYCLIC&1680-00-018-9343&209-001-320-5&PAOFZ&
97499&EA&1&BITMAPS\AH246D.BMP&
32 BEARING,PLANE,SELF-LOCK&3120-00-931-4788&209-001-053-3&
PAFZZ&97499&EA&1&NONE&

5.FUELSYS.DAT *(Fuel System)*

1 TANK ASSEMBLY,FUEL&1560-01-018-5919&209-060-626-3&PAFDD&
97499&EA&1&BITMAPS\AH242E.BMP&
2 TANK ASSEMBLY,AIRCRAFT&1560-01-018-4096&209-060-626-1&PAFDD&
97499&EA&1&BITMAPS\AH242E.BMP&
3 FITTING ASSEMBLY&NONE&FCD56297&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
3A RING, FUEL CELL&NONE&FCD55036&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
3B INSERT, SCREW THREADED&5340-00-829-2141&3591-4CN0375&PAFZZ&
01556&EA&1&BITMAPS\AH242E.BMP&
4 FITTING ASSEMBLY&NONE&FCD56291&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
4B INSERT, SCREW THREADED&5340-00-829-2141&3591-4CN0375&PAFZZ&
01556& EA&18&BITMAPS\AH242E.BMP&
5 FITTING ASSEMBLY&NONE&FCD56290&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
5A RING, FUEL CELL&NONE&FCD56200&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
5B INSERT, SCREW THREADED&5340-00-829-2141&3591-4CN0375&PAFZZ&
01556& EA&10&BITMAPS\AH242E.BMP&
6 FITTING ASSEMBLY&NONE&FCD55333&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
6A RING, FUEL CELL&NONE&FCD55334&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
6B INSERT, SCREW THREADED&5340-00-829-2141&3591-4CN0375&PAFZZ&
01556&EA&12&BITMAPS\AH242E.BMP&
7 FITTING ASSEMBLY&NONE&FCD55321&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
7A RING, FUEL CELL&NONE&FCD55322&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
7B INSERT, SCREW THREADED&5340-00-829-2141&3591-4CN0375&PAFZZ&
01556&EA&4&BITMAPS\AH242E.BMP&
8 FITTING ASSEMBLY&NONE&FCD56449&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
8A RING, FUEL CELL&NONE&FCD56448&XAFZZ&00333&EA&1&

BITMAPS\AH242E.BMP&
8B INSERT, SCREW THREADED&5340-00-829-2141&3591-4CN0375&PAFZZ&
01556&EA&4&BITMAPS\AH242E.BMP&
9 FITTING ASSEMBLY&NONE&FCD56318&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
9A RING, FUEL CELL&NONE&FCD56317&XAFZZ&00333&EA&1&
BITMAPS\AH242E.BMP&
9B INSERT, SCREW THREADED&5340-00-829-2141&3591-4CN0375&PAFZZ&
01556&EA&2&BITMAPS\AH242E.BMP&
FILTER,FLUID,PRESSURE BKDOWN&2915-00-003-5904&204-040-760-5&
PAOOO&97499&EA&1&BITMAPS\AH237D.BMP&
1 HEAD, FLUID FILTER&2915-00-868-2708&204-040-760-11&
PAOOO&97499&EA&1&BITMAPS\AH237D.BMP&
2 SCREW, MACHINE&5305-00-967-7712&126221&PAOZZ&73370&
EA&4&BITMAPS\AH237D.BMP&
2 SCREW,SELF LOCKING&5305-00-720-8562&NK500C8-8&PAOZZ&02615&
EA&4&BITMAPS\AH237D.BMP&
3 SWITCH, PRESSURE&5930-00-961-1318&100316&PAOZZ&05160&
EA&4&BITMAPS\AH237D.BMP&
4 PACKING,PREFORMED&5330-00-841-2366&MS29513-030&
PCOZZ&96906&EA&2&BITMAPS\AH237D.BMP&
5 COUPLING,CLAMP&5340-00-225-9053&4562-350&PAOZZ&
98625&EA&1&BITMAPS\AH237E.BMP&
5A NUT, SELF LOCKING&5310-00-813-3232&NAS679C4M&PAOZZ&80205&
EA&1&BITMAPS\AH237E.BMP&
6 STRAINER BODY,SEDIM&1560-00757-8745&204-040-760-15&PBOZZ&
97499&EA&1&BITMAPS\AH237E.BMP&
7 FILTER ELEMENT&NONE&204-040-760-13&XAOZZ&97499&EA&1&
BITMAPS\AH237D.BMP&
8 PACKING,PREFORMED&5330-00-250-0236&MS29513-024&
PCOZZ&96906&EA&2&BITMAPS\AH237D.BMP&
FUEL AND OIL KIT FILTER&2945-00-019-0280&204-2490-1&PAOZZ&
97499&EA&1&NONE&

6.SPTOOLS.DAT (Special Tools)

1 WRENCH,SPANNER-1&AUTH/21/50 EQUIP&5120-00-837-9483&T101493&
XBOZZ&97499&EA&2&BITMAPS\AHA1B.BMP&
3 TRANSMISSION ADAPTOR-1& AUTH/21/50 EQUIP&4920-00-676-2307&
SWE13852-40&PADZZ&87641&EA&1&BITMAPS\AHA1B.BMP&
4 THUMBSCREW-1& AUTH/21/50 EQUIP&5303-00-765-4469&TLOO929&
PAFZZ&97499&EA&1&BITMAPS\AHA1C.BMP&
5 WRENCH,INPUT QUILL&AUTH/21/50 EQUIP&5120-00-932-3670&
T101488&PBFZZ&97499&EA&1&BITMAPS\AHA1C.BMP&
8 SOCKET,SPLINED REATION&AUTH/21/50 EQUIP&5120-00-619-9774&
PD2657&PBDZZ&81996&EA&1&BITMAPS\AHA1A.BMP&
9 SOCKET,SPLINED &AUTH/21/50 EQUIP&5120-00-619-9773&PD2658&

PBFZZ&81996&EA&V&BITMAPS\AHA1A.BMP&
10 SOCKET,MASK NUT &N/A&5120-00-619-9779&PD2659&
PBOZZ&81996&EA&V&BITMAPS\AHA1A.BMP&
11ADAPTOR, REACTION &N/A&5120-00-619-9776&PD2660&PBOZZ&
81996&EA&V&BITMAPS\AHA1A.BMP&
1 BAR,BEARING REMOVAL&N/A&4920-00-474-0567&T101333&PADZZ&
97499&EA&2&BITMAPS\AHA3A.BMP&
1 BAR,DUPLEX BEARING&N/A&4920-00-718-0567&T101333&PADZZ&97499&

EA&2&BITMAPS\AHA3A.BMP&
2 PLATE ASSEMBLY& AUTH/21/50 EQUIP&4920-00-967-7651&
SWE13852-40&PADZZ&87641&EA&1&BITMAPS\AHA3A.BMP&
4 BAR,DUPLEX BEARING REMOVAL&N/A&4920-00-718-6540&T101337&
PADZZ&97499&EA&2&BITMAPS\AHA3A.BMP&
5 JACK, SCREW, SET& AUTH/21/50 EQUIP&4920-00-765-4410&T1O1338&
PAFZZ&97499&SE&1&BITMAPS\AHA3A.BMP&
6 FIXTURE, RIGGING&AUTH/21/50 EQUIP&4920-00-848-4930&T101524&
PBOBZ&97499&EA&1&BITMAPS\AHA3A.BMP&
6A PIN, QUICK RELEASE&AUTH/21/50 EQUIP&5340-00-071-8267&
LW1129-1-900&PAOZZ&83014&EA&1&BITMAPS\AHA3A.BMP&
6A BALL,RIGGING FIXTURE &AUTH/21/50 EQUIP&5340-00-071-8267&
50454&PAOZZ&81240&EA&V&BITMAPS\AHA3A.BMP&
6B NUT,PLAIN WING&AUTH/21/50 EQUIP&5310-00-063-6716&
MS35426-14&PAOZZ&96906&EA&V&BITMAPS\AHA3A.BMP&

1 GRIP,SPACING AND BALANCING&AUTH/21/50 EQUIP&
4920-00-251-2509&T101457&PBFZZ&97499&EA&1&BITMAPS\AHA61.BMP&
2 SOCKET,WRENCH,FACE& AUTH/21/50 EQUIP&5120-00-044-1426&
T101414&XBOZZ&97499&EA&1&BITMAPS\AHA62.BMP&
3 FIXTURE,HOLDING,SHAFT&AUTH/21/50 EQUIP&4920-00-876-0103&
T101420&PBOZZ&97499&EA&1&BITMAPS\AHA63.BMP&
3 PULLER, BEARING& AUTH/21/50 EQUIP&5120-00-999-5306&
T101491&PBFZZ&97499&EA&1&NONE&
3 CROWFOOT,ATTACHMENT& AUTH/21/50 EQUIP&5120-00-184-8413&
GGGW641&PBOZZ&97499&EA&V&NONE&
4 TOOL SET, ALIGNMENT& AUTH/21/50 EQUIP&5120-00-894-0014&
T101419&PBOZZ&97499&EA&1&BITMAPS\AHA64.BMP&
5 TARGET ASSEMBLY& AUTH/21/50 EQUIP&4920-00-898-0013&
T101419-5&PBOZZ&97499&EA&1&BITMAPS\AHA64.BMP&
6 WRENCH& N/A&5120-00-018-0220&
T101456&PAOFZZ&97499&EA&1&BITMAPS\AHA65.BMP&

APPENDIX E. AH1S TEMPLATE FILES

/* template file for the cobra object*/
1.COBRA.TPL File

```
1@1
1@2
20@2
30@20
```

/* template file for the electrical system object */
2.ELECSYS.TPL File

```
1@1
1@2
30@1
1@5
1@6
16@1
1@9
1@10
25@1
1@13
1@14
5@1
10@13
10@14
5@1
1@17
1@18
5@1
10@17
10@18
5@1
40@29
```

/* template file for the environmental system object */
3.ENVSYS.TPL File

```
1@1
1@2
30@1
1@5
1@6
16@1
1@9
```

```
1@10
25@1
1@13
1@14

5@1
10@13
10@14
5@1
1@17
1@18
5@1
10@17
10@18
5@1
40@29
```

/* template file for the flight control system object */
4.FCTLSYS.TPL File

```
1@1
1@2
30@1
1@5
1@6
16@1
1@9
1@10
25@1
1@13
1@14
5@1
10@13
10@14
5@1
1@17
1@18
5@1
10@17
10@18
5@1
40@29
```

/* template file for the fuel system object */
5.FUELSYS.TPL File

```
1@1
```

```
1@2
30@1
1@5
1@6
16@1
1@9
1@10
25@1
1@13
1@14
5@1
10@13
10@14
5@1
1@17
1@18
5@1
10@17
10@18
5@1
40@29
```

/* template file for the special tools object */
6.SPTOOLS.TPL File

```
1@1
1@2
30@1
1@5
1@6
25@1
1@9
1@10
16@1
1@13
1@14
25@1
1@17
1@18
5@1
10@17
10@18
5@1
1@21
1@22
5@1
10@21
```

10@22
5@1
40@40

APPENDIX F. STAND-ALONE FILES

/* Files reguired to compile the stand-alone application.*/

1.ACTGLAD.LOD File

```
LoadFiles := #("glad.h",
              "act\colortab.act",
              "act\gladnil.act",
              "act\gladstr.act",
              "act\literals.act",
              "classes\textfile.cls",
              "classes\string.cls",
              "classes\long.cls",
              "classes\gladobj.cls",
              "classes\dbdialog.cls",
              "classes\typedial.cls"
              "classes\attribdi.cls",
              "classes\mywindow.cls",
              "classes\control.cls",
              "classes\edit.cls",
              "classes\tempomem.cls",
              "classes\tempdow.cls",
              "classes\onememwindow.cls",
              "classes\listmemw.cls",
              "classes\dispedit.cls",
              "classes\button.cls",
              "classes\tempbm.cls",
              "classes\tempdelw.cls",
              "classes\tempaddo.cls",
              "classes\displayone.cls",
              "classes\bitmaps.cls",
              "classes\colortab.cls",
              "classes\objectdi.cls",
              "classes\tempdmw.cls",
              "classes\tempndmw.cls",
              "classes\describe.cls",
              "classes\dmwindow.cls",
              "classes\ddwindow.cls",
              "classes\nestdmwi.cls",
              "classes\tempapp.cls",
              "classes\g!adwind.cls",
              "classes\addonewi.cls",
              "classes\deletewi.cls",
              "act\install.act",
              "act\gladapp.act")!!
```

2.GLADAPP.ACT File

```
/* This file is used for installing the GLAD application. */

/* Define the application class */
inherit(Object, #GladApp, #( display /* GLAD main window*/ ),
nil, nil)!!

/* Define an init method for GladApp. This is executed when
the
 application starts up, and must create the windows and
objects
 necessary for the application. */!!

now(GladApp)!!

Def init(self | openDlg)
{       initSystem(self);

/* create and show the application window. This assumes that
 the new method takes 0 args. */
        display := new(GladWindow,ThePort,"GladTopMenu","G L
A D",nil);
          display.isMain := true;
        show(display, 3);
        openDlg := new(Dialog);
        runModal(openDlg,ABOUT_GLAD,display);
}!!

/* If any cleaning up needs to be done in the application
before closing,
it should be done here. */
Def shouldClose(self)
{}!!

/* The installation method for the application. First dispose

   of classes and methods needed only for compilation. Then
run    a cleanup to get rid of the static objects, and save
the image    with memory values. Finally, exit because Actor
can't run        without the compiler.
 Note: This method must be invoked via abort(installUser)),

        or the static garbage collection will fail to reclaim
the         compiler memory because the parser is accessible
```

```
on the          stack. */!!
now(Object)!!


Def installGladApp(self)
{        setName(VImage, "GLADV02.ima");
         removeCompiler();
/*  removeJunk is a user-defined method in which you can
remove
    unneeded classes and methods */
/*       removeJunk();
         remove(Object.methods, #removeJunk);
*/
         TheApp := new(GladApp);
         cleanup();
         create(VImage);
         snap(VImage, 100, 27);          /* 110K static, 27K
                                        dynamic */
         close(VImage);
         exit();
}!!
```

3. GLAD.H file

```
/* GLAD constants definitions */

#define SHADE_BOR_WD   20
#define SHADE_BOR_HT   10
#define LABEL_LOC       0
#define EC_LOC          1
#define EC_DIM          2
#define COLOR           0
#define USED            1
#define NAME            0
#define CLASS           1
#define LENGTH          2
#define USER_DEF        3
#define DEFBUTTON       1
#define CREATE_DB     800
#define MODIFY_DB     801
#define OPEN_DB       802
#define REMOVE_DB     803
#define ABOUT_DB      815
#define ABOUT_GLAD    900
#define OPNDBLIST     910
```

```
#define RMVDBLIST     911
#define DB_LB        912
#define HELP_LB       913
#define DEFOBJ       920
#define OBJ_NAME      921
#define ATOMIC        922
#define NESTED        923
#define LEVEL        924
#define ATTRIB       930
#define ATTR_NAME     931
#define ATTR_TYPE     932
#define ATTR_LENGTH   933
#define ATTR_LIST     934
#define ATTR_DELETE   935
#define TYPE_LIST     936
#define ATTRLIST      940
#define DT_CENTER       1
#define DT_VCENTER      4
#define DT_SINGLELINE   32
#define WHITE_COLOR     0xFFFFFFL
#define ES_LEFT        0L
#define ES_MULTILINE    4L
#define DSTINVERT       0x00550009L
#define MF_DELETE       0x0200
#define HELPER        950
!!
```

4. GLADV02.RC File

```
; This file is the resource(.RC) file for the GLAD
application.

#include "style.h"
#include "actor.h"
#include "glad.h"

; include your application's icon here
  work    ICON  glad.ico

STRINGTABLE
BEGIN
; substitute your application name in the next two strings.
        IDSNAME, "GLADV02"
        IDSAPP,  "GLADV02.IMA"

        dosError, " reported DOS error# "
```

```
; Used for results of checkError
        52, "File not found"
        53, "Path not found"
        54, "No handle available; all in use"
        55, "Access denied"
        65, "Invalid drive specification"

; various fatal error strings - should be kept
        150, "Attempted to move freed object:"
        152, "Dynamic memory is full."
        153, "Free list is corrupted."
        154, "Scavenge list is full."
        155, "Out of object pointers."
        157, "Snapshot load failed."
        158, "Not enough memory to run Actor."
        160, "Actor Display"
        161, "Requires higher static setting."
        162, "Requires higher dynamic setting."
        163, "ACTOR Version 1.0"
        164, "Windows/Actor stack overflowed  "
        165, "Windows/Actor stack underflowed "
        166, "Actor stack overflowed"
        167, "Corrupted object memory"
        168, "Actor symbol table is full"

END

; use the following for your own accelerators
; use your application name in place of Actor -- acr
GLADV02 ACCELERATORS
BEGIN
 VK_INSERT, EDIT_PASTE, VIRTKEY
 VK_SUBTRACT, EDIT_CUT, VIRTKEY
 VK_ADD, EDIT_COPY, VIRTKEY

 VK_LEFT, VK_LEFT, VIRTKEY
 VK_UP, VK_UP, VIRTKEY
 VK_RIGHT, VK_RIGHT, VIRTKEY
 VK_DOWN, VK_DOWN, VIRTKEY

 "^a", EDIT_SELALL
 "^r", BR_REFORM
 "^z", BR_ZOOM

 VK_F1, HELPER, VIRTKEY
 VK_TAB, EDIT_TAB, VIRTKEY
```

```
  VK_PRIOR, EDIT_PRIOR, VIRTKEY
  VK_NEXT, EDIT_NEXT, VIRTKEY
  VK_HOME, EDIT_HOME, VIRTKEY
  VK_END, EDIT_END, VIRTKEY

  VK_DELETE, EDIT_CLEAR, VIRTKEY
  VK_DELETE, EDIT_CUT, VIRTKEY, SHIFT
  VK_INSERT, EDIT_COPY, VIRTKEY, CONTROL
  VK_INSERT, EDIT_PASTE, VIRTKEY, SHIFT
END


; your menus can go here
EditMenu  MENU
BEGIN
  POPUP "&Edit"
    BEGIN
    MENUITEM "Cu&t\tShift+Del",  EDIT_CUT
    MENUITEM "&Copy\tCtrl+Ins", EDIT_COPY
    MENUITEM "&Paste\tShift+Ins", EDIT_PASTE
    MENUITEM "C&lear\tDel", EDIT_CLEAR
    END
END


GladTopMenu  MENU
BEGIN
  MENUITEM  "Create", 1
  MENUITEM  "Modify", 2
  MENUITEM  "Open",   3
  MENUITEM  "Remove", 4
  MENUITEM  "Quit",   6
  MENUITEM  "\aF1=Help", HELPER, HELP
END


GladDmlMenu  MENU
BEGIN
  MENUITEM  "Describe",  1
  MENUITEM  "Expand",  2
  POPUP "ListMembers"
  BEGIN
 MENUITEM "All at Once",  3
 MENUITEM "One by One",  4
  END
  POPUP "Change"
  BEGIN
 MENUITEM "Add data",  5
 MENUITEM "Delete data",6
  MENUITEM "Modify data", 7
  END
  MENUITEM  "Query",       8
  MENUITEM  "ShowConnection", 9
  MENUITEM  "Quit",        11
  MENUITEM  "\aF1=Help", HELPER, HELP
END


GladDdlMenu  MENU
BEGIN
  MENUITEM  "Save",   1
  MENUITEM  "Define",  2
  MENUITEM  "Attribute", 3
  MENUITEM  "Expand",  4
  MENUITEM  "Delete",  5
  MENUITEM  "Quit",    7
  MENUITEM  "\aF1 Help", HELPER,HELP
END


GladLMMenu  MENU
BEGIN
  MENUITEM  "More",   1
  MENUITEM  "Modify", 2
  MENUITEM  "Quit",   4
  MENUITEM  "\aF1=Help",  HELPER,HELP
END

GladOMMenu  MENU
BEGIN
 MENUITEM  "Add",    1
 MENUITEM  "Delete", 2
 MENUITEM  "Modify", 3
 MENUITEM  "Prev",   4
 MENUITEM  "Next",  5
 POPUP  "GoTo"
 BEGIN
 MENUITEM   "First",  6
 MENUITEM   "Last",   7
  MENUITEM   "I th",   8
 END
 MENUITEM  "All",  9
 MENUITEM  "Quit", 11
 MENUITEM  "\aF1=Help", HELPER,HELP
```

```
END


GladCOMenu MENU
BEGIN
  MENUITEM "Quit", 1
END

ABOUT_GLAD  DIALOG  90,34,122,80
STYLE  WS_DLGFRAME  | WS_POPUP
BEGIN
  CTEXT  "GLAD Version 0.02",  -1, 23,12,72,11, WS_CHILD
  CTEXT  "Naval Postgraduate School", -1, 8,25,105,10,WS_CHILD

  CTEXT  "Dept of Computer Science", -1, 9,37,100,11, WS_CHILD

  ICON  "work",-1,26,50,16,16, WS_CHILD
  DEFPUSHBUTTON "START", IDOK, 70,58,39,14, WS_CHILD
END

OPNDBLIST DIALOG LOADONCALL MOVEABLE DISCARDABLE 70, 23, 166,

102
CAPTION "GLAD  Databases"
STYLE WS_DLGFRAME | WS_POPUP
BEGIN
   CONTROL "" DB_LB, "listbox", LBS_NOTIFY | LBS_SORT |
LBS_STANDARD | WS_BORDER | WS_VSCROLL | WS_CHILD, 5, 16, 110,
82
   CONTROL "OPEN" DEFBUTTON, "button", BS_DEFPUSHBUTTON |
WS_TABSTOP | WS_CHILD, 125, 17, 33, 13
   CONTROL "ABOUT" ABOUT_DB, "button", BS_PUSHBUTTON |
WS_TABSTOP | WS_CHILD, 125, 41, 33, 13
   CONTROL "HELP" HELP_LB, "button", BS_PUSHBUTTON |
WS_TABSTOP | WS_CHILD, 126, 62, 32, 13
   CONTROL "CANCEL" 2, "button", BS_PUSHBUTTON | WS_TABSTOP
| WS_CHILD, 125, 82, 33, 13
   CONTROL "GLAD  Databases" -1, "static", SS_CENTER |
WS_CHILD, 17, 4, 83, 10
END

RMVDBLIST DIALOG LOADONCALL MOVEABLE DISCARDABLE 70, 23, 166,

102
```

```
CAPTION "GLAD  Databases"
STYLE WS_DLGFRAME | WS_POPUP
BEGIN
   CONTROL "" DB_LB, "listbox", LBS_NOTIFY | LBS_SORT |
LBS_STANDARD | WS_BORDER | WS_VSCROLL | WS_CHILD, 5, 16, 115,
82
   CONTROL "REMOVE" DEFBUTTON, "button", BS_DEFPUSHBUTTON |
WS_TABSTOP | WS_CHILD, 126, 16, 33, 13
   CONTROL "CANCEL" 2, "button", BS_PUSHBUTTON | WS_TABSTOP
| WS_CHILD, 126, 81, 33, 13
   CONTROL "ABOUT" ABOUT_DB, "button", BS_PUSHBUTTON |
WS_TABSTOP | WS_CHILD, 126, 39, 33, 13
   CONTROL "HELP" HELP_LB, "button", BS_PUSHBUTTON |
WS_TABSTOP | WS_CHILD, 127, 61, 32, 13
   CONTROL "SELECT the one to be REMOVED" -1, "static",
SS_CENTER | WS_CHILD, 0, 3, 124, 10
END


DEFOBJ DIALOG LOADONCALL MOVEABLE DISCARDABLE 23, 21, 136, 98

CAPTION "OBJECT DEFINITION"
STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME | WS_POPUP
BEGIN
    CONTROL "Enter Object Name:" 0, "static", SS_LEFT |
WS_CHILD, 8, 5, 74, 10
    CONTROL "" OBJ_NAME, "edit", ES_LEFT | WS_BORDER |
WS_TABSTOP | WS_CHILD, 8, 16, 117, 12
    CONTROL "Atomic" ATOMIC, "button", BS_RADIOBUTTON |
WS_GROUP | WS_TABSTOP | WS_CHILD, 25, 44, 40, 12
    CONTROL "Nested" NESTED, "button", BS_RADIOBUTTON |
WS_TABSTOP | WS_CHILD, 70, 44, 41, 12
    CONTROL "Nesting Level" LEVEL, "button", BS_GROUPBOX |
WS_TABSTOP | WS_CHILD, 20, 31, 93, 30
    CONTROL "Accept" IDOK, "button", BS_PUSHBUTTON | WS_GROUP
| WS_TABSTOP | WS_CHILD, 17, 70, 42, 14
    CONTROL "Cancel" IDCANCEL, "button", BS_PUSHBUTTON |
WS_TABSTOP | WS_CHILD, 76, 71, 42, 14
END

ATTRIB DIALOG LOADONCALL MOVEABLE DISCARDABLE 11, 18, 208, 216


STYLE WS_DLGFRAME | WS_POPUP
BEGIN
          CONTROL "Attribute Name:" DT_CENTER, "static",
SS_LEFT | WS_CHILD, 6, 18, 64, 12
```

```
        CONTROL "Attribute Type:" 5, "static", SS_LEFT |
WS_CHILD, 6, 54, 79, 12
        CONTROL "Length of field:" 15, "static", SS_LEFT
| WS_CHILD, 6, 90, 86, 10
        CONTROL "" ATTR_NAME, "edit", ES_LEFT | WS_BORDER
| WS_TABSTOP | WS_CHILD, 5, 30, 105, 15
        CONTROL "" ATTR_TYPE, "edit", ES_LEFT | WS_BORDER
| WS_TABSTOP | WS_CHILD, 5, 67, 105, 15
        CONTROL "" ATTR_LENGTH, "edit", ES_LEFT | WS_BORDER
| WS_TABSTOP | WS_CHILD, 5, 102, 105, 16
        CONTROL "" ATTR_LIST, "listbox", LBS_NOTIFY |
LBS_SORT | LBS_STANDARD | WS_BORDER | WS_VSCROLL | WS_CHILD,
5, 126, 106, 82
        CONTROL "Add" IDOK, "button", BS_PUSHBUTTON |
WS_TABSTOP | WS_CHILD, 138, 42, 44, 14
        CONTROL "Delete" ATTR_DELETE, "button",
BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 138, 72, 44, 14
        CONTROL "Type List" TYPE_LIST, "button",
BS_PUSHBUTTON | WS_TABSTOP | WS_CHILD, 138, 102, 44, 15
        CONTROL "Quit" IDCANCEL, "button", BS_PUSHBUTTON
| WS_TABSTOP | WS_CHILD, 139, 132, 44, 14
        CONTROL "Attributes for object" 16, "static",
SS_LEFT | WS_CHILD, 29, 5, 86, 8
        CONTROL "" OBJ_NAME, "edit", ES_LEFT | WS_TABSTOP
| WS_CHILD, 118, 5, 74, 12
END

ATTRLIST DIALOG LOADONCALL MOVEABLE DISCARDABLE 11, 18, 122,
122
STYLE WS_DLGFRAME | WS_POPUP
BEGIN
   CONTROL "" TYPE_LIST, "listbox", LBS_NOTIFY | LBS_SORT |
LBS_STANDARD | WS_BORDER | WS_VSCROLL | WS_CHILD, 7, 6, 105,
74
   CONTROL "Accept" IDOK, "button", BS_PUSHBUTTON |
WS_TABSTOP | WS_CHILD, 18, 90, 33, 14
   CONTROL "Cancel" IDCANCEL, "button", BS_PUSHBUTTON |
WS_TABSTOP | WS_CHILD, 68, 90, 29, 14
END

INPUT_BOX DIALOG DISCARDABLE 77, 94, 165, 71
STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME | WS_POPUP
BEGIN
    EDITTEXT FILE_EDIT, 10, 32, 138, 12, WS_BORDER | WS_CHILD
| WS_TABSTOP |
    ES_AUTOHSCROLL
    LTEXT " ", INPUT_MSG, 11, 5, 143, 18, WS_CHILD
    DEFPUSHBUTTON "&OK", IDOK, 32, 50, 32, 14, WS_CHILD
    PUSHBUTTON "&Cancel" IDCANCEL, 99, 50, 32, 14, WS_CHILD
END
```

LIST OF REFERENCES

1. Moreau, James G., "CSS Automation: An Overview," Army Logistician, January-February, pp. 22-24, 1989.

2. Chase, James and Rogowski,Joseph J., "The Paperless Manual," Army Logistician, September-October, pp. 41-42, 1987.

3. Shneiderman, Ben, Designing the User Interface: Strategies for Effective Human-Computer Interaction, Addison-Wesley, Reading, Massachusetts, 1987, 14-41.

4. Meyer, Bertrand, Object-oriented Software Construction, Printice Hall International(UK) Ltd, Cambridge, Great Britain, 1988.

5. Cox, Brad J., Object Oriented Programming: An Evolutionary Approach, Addison-Wesley, Reading, Massachusetts, 1987.

6. Duff, Charles, et.al., Actor Language Manual, The Whitewater Group, Evanston, Illinois, 1987.

7. Duff, Charles, et.al., Actor Training Course Manual, Version 1.1, The Whitewater Group, Evanston, Illinois, February 1988.

8. Schuett, Robert J., Prototyping Visual Database Interface by Object Oriented Design, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1988, 5-13.

9. Microsoft Corporation, Microsoft Windows 2.0 Software Development Kit, Microsoft Press, 1988.

10. Naval Postgraduate School Technical Report NPS52-88-050, Implementing Visual Database Interface By Using Object-Oriented Language, by C. Thomas, Wu and David K. Hsiao, September 1988.

11. Williamson, Michael L., An Implementation of a Data Definition Facility for the Graphics Language for Database, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1988, 21-35.

12. Technical Manual 55-1520-221-23P-1, Aviation Unit and Intermediate Maintenance Repair Parts and Special Tools List, Change 1.0, Headquarters Department of the Army, Washington, D.C., April 1980.

13. Korth, Henry K. and Silberschatz, Abraham, Database System Concepts, McGraw-Hill Book company, New York, New York, 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Director, Project BASIS Naval Data Automation Command Washington Navy Yard Washington, D.C. 20374	1
4.	Curriculum Officer Computer Technology Program, Code 37 Monterey, California 93943-5000	1
5.	Professor C. Thomas Wu, Code 52Wu Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	10
6.	Colonel James G. Moreau, USA(Ret) Deputy Commander for Logistics Automation Army Logistics Center Fort Lee, Virginia 23801-6145	1
7.	CPT(P) Henry R. Fore 88-89 CGSOC Regular Course C/O Post Locator Fort Leavenworth, Kansas 66027-7200	2